UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

PARADIGMA: UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE MODELO CONCEITUAL DE CLASSES BASEADA EM PROCESSAMENTO DE LINGUAGEM NATURAL

WILSON CARLOS DA SILVA

ORIENTADOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

PIRACICABA, SP
2008

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

PARADIGMA: UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE MODELO CONCEITUAL DE CLASSES BASEADA EM PROCESSAMENTO DE LINGUAGEM NATURAL

WILSON CARLOS DA SILVA

ORIENTADOR: PROF. DR. LUIZ EDUARDO GALVÃO MARTINS

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP
2008

Silva, Wilson Carlos da

PARADIGMA: uma ferramenta para geração automática de modelo conceitual de classes baseada em Processamento de Linguagem Natural, 2008.

Orientador: Prof. Dr. Luiz Eduardo Galvão Martins.

Dissertação (Mestrado) - Programa de Pós-Graduação em Ciência da Computação – Faculdade de Ciências Exatas e da Natureza - Universidade Metodista de Piracicaba - UNIMEP.

1- Engenharia de Requisitos. 2- Processamento de Linguagem Natural.
3- Orientação a Objetos. 4- Elicitação de Requisitos
5- Modelagem de Requisitos.

PARADIGMA: UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE MODELO CONCEITUAL DE CLASSES BASEADA EM PROCESSAMENTO DE LINGUAGEM NATURAL

Autor: Wilson Carlos da Silva

Orientador: Prof.Dr. Luiz Eduardo Galvão Martins

Dissertação de Mestrado defendida e aprovada em 22 de fevereiro de 2008, pela banca examinadora constituída dos Professores:

Prof. Dr. Luiz Eduardo Galvão Martins
UNIMEP

Profa. Dra. Marina Teresa Pires Vieira
UNIMEP

Profa. Dra. Miriam Sayão
PUC-RS

À

Minha esposa Thais Michelli e aos meus filhos Fábio e Daniel.

Aos meus pais Luzia e Eliseu.

AGRADECIMENTOS

Agradeço a Deus pelas bênçãos que tenho recebido e pela condução da minha vida até este momento.

Agradeço à minha esposa Thais Michelli e aos meus filhos Fábio e Daniel pelo amor, carinho, dedicação e compreensão nos momentos que estive ausente estudando.

Agradeço aos meus pais e irmãos pelo incentivo que sempre me deram para que eu trilhasse o caminho do aprender.

Agradeço ao meu orientador, Prof. Dr. Luiz Eduardo Galvão Martins, pelo incentivo, apoio e orientações durante a realização deste trabalho e por confiar no meu potencial.

Agradeço aos professores do programa de Mestrado em Ciência da Computação da UNIMEP pelo conhecimento compartilhado e incentivo à pesquisa.

Agradeço aos amigos Carlos, Clodonil e Edmar que sempre me motivaram nas infindas viagens de São Paulo a Piracicaba, pois estávamos em busca dos mesmos ideais.

Agradecimentos finais aos diretores do Centro Universitário Adventista de São Paulo (UNASP-SP) pelo apoio irrestrito nessa etapa tão importante da minha vida: tanto apoio financeiro, como na liberação nos dias necessários para realização deste trabalho.

"Tudo tem seu tempo e até certas manifestações mais vigorosas e originais, entram em voga ou saem de moda. Mas a sabedoria tem uma vantagem: é eterna". Baltasar Gracián

RESUMO

Os engenheiros de requisitos contam com várias técnicas que os auxiliam no processo de elicitação, análise, especificação, validação e gerenciamento de requisitos, mas o número de ferramentas nesse segmento ainda é reduzido. Os requisitos são a base para o desenvolvimento do software, os quais, na maioria das vezes, são descritos em linguagem natural. Neste trabalho é apresentada uma ferramenta chamada PARADIGMA, desenvolvida para auxiliar os engenheiros de requisitos a identificarem, com maior facilidade, classes, atributos, operações e relacionamentos a partir dos requisitos descritos em linguagem natural, utilizando-se do processamento da linguagem natural (mais especificamente os etiquetadores morfossintáticos), gerando um modelo conceitual de classes da UML (*Unified Modeling Language*). Além da característica mencionada acima, a ferramenta utiliza o conceito de padrões lingüísticos, que facilitam a criação de modelos mais próximos daqueles criados por modelador humano. A experimentação da ferramenta foi realizada por profissionais e professores que atuam na área de Engenharia de Requisitos. Com base nos resultados, podemos constatar a relevância da ferramenta PARADIGMA no contexto da Engenharia de Requisitos.

PALAVRAS-CHAVE: Engenharia de Requisitos, Processamento de Linguagem Natural, Orientação a Objetos, Elicitação de Requisitos, Modelagem de Requisitos.

ABSTRACT

The requirements engineers count on vary techniques that assist them on the elicitation, analysis, specification, validation and requirements management process, but the number of those tools in this segment is still reduced. The requirements are the base for the software development, which, most of time is described in natural language. In this work a tool called PARADIGMA is presented, it was developed to assist the requirements engineers to identify in a easier way classes, attributes, operations and relationships from the described requirements in natural language, and using the Natural Language Processing (more specifically the morphosyntatics taggers), generating a conceptual classes model of UML (Unified Modeling Language). Besides the mentioned characteristics above, the tool uses the concept of linguistic standards that facilitate the creation of diagrams closer to those created by human. The experimentation of the tool was carried through by professional and professors who act in the area of Requirements Engineering. On the basis of the results, we come to the conclusion of the relevance of the tool PARADIGMA in the context of the Requirements Engineering.

KEY WORDS: Requirements Engineering, Natural Language Processing, Object Oriented, Requirements Elicitation, Requirements Modeling.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

LN	Linguagem Natural
PLN	Processamento de Linguagem Natural
UML	*Unified Modeling Language*
CASE	*Computer Aided Software Engineering*
RUP	*Rational Unified Process*
LIDA	*Linguistic Assistant for Domain Analysis*
UCDA	*Use Case driven Development Assistant*
POS	*Part-of-speech*
MXPOST	*Maximum Entropy POS tagger*
DFD	Diagrama de Fluxo de Dados
DER	Diagrama Entidade Relacionamento
VDM	*Vienna Development Method*
CSP	*Communicating Sequencial Processes*
CM-Builder	*Class Model Builder*
UdI	*Universo de Informações*

LISTA DE TABELAS

1 INTRODUÇÃO

A base para a construção de um software é organizada pela Engenharia de Requisitos. A Engenharia de Requisitos dá suporte à fase inicial do ciclo de vida do software. De acordo com Kotonya e Sommerville (1998), a Engenharia de Requisitos engloba todas as atividades envolvidas na descoberta, na documentação e manutenção de um conjunto de requisitos para um sistema computacional.

A especificação de requisitos é um documento, ou artefato, utilizado para a comunicação entre os clientes, usuários, engenheiros de requisitos e desenvolvedores de software. A maioria das especificações de requisitos é escrita em linguagem natural, sendo complementada, às vezes, por outros tipos de notações, tais como diagramas, fórmulas, modelos formais. A especificação de requisitos em linguagem natural facilita a comunicação humana, pois podemos descrever as minúcias dos problemas e suas respectivas soluções.

De acordo com uma pesquisa *on-line* conduzida pela Universidade de Trento (MICH; FRANCH; NOVI INVERARDI, 2003), a maioria dos documentos disponíveis para análise dos requisitos está em linguagem natural ou é constituída de informações obtidas através de entrevistas. Conseguiram mediante essa pesquisa chegar aos seguintes números:

- 79% dos documentos são escritos em linguagem natural comum;
- 16% dos documentos são escritos em linguagem natural estruturada;
- e somente 5% dos documentos são escritos em linguagem formal.

De acordo com os dados apresentados nessa pesquisa, fica evidente a grande quantidade de documentos em linguagem natural no processo de elicitação de requisitos. Existem várias técnicas que auxiliam os engenheiros de requisitos na elicitação dos requisitos, para que posteriormente os requisitos possam ser especificados da melhor maneira possível. Segundo Davis e Hickey (2003), essas técnicas são utilizadas pelos engenheiros de requisitos para determinar as necessidades dos clientes e usuários. Essas necessidades, sendo identificadas claramente, poderão levar à construção de um sistema com grande chance de atender às expectativas e às reais necessidades dos usuários.

1.1 OBJETIVO DO TRABALHO

Os engenheiros de requisitos contam com várias técnicas que os auxiliam no processo de elicitação, análise, especificação, validação e gerenciamento de requisitos. Os requisitos são a base para o desenvolvimento do software, os quais, na maioria das vezes, são descritos em linguagem natural.

O objetivo deste trabalho é desenvolver uma ferramenta para auxiliar os engenheiros de requisitos a identificarem, com maior facilidade, classes, atributos, operações e associações a partir dos requisitos descritos em linguagem natural, utilizando-se do processamento da linguagem natural, gerando um modelo conceitual de classes da UML (*Unified Modeling Language*). Além desta característica, a ferramenta utilizará padrões lingüísticos pré-configurados, que facilitam a criação de modelos mais próximos daqueles criados por modelador humano.

A ferramenta irá gerar o modelo conceitual de classes de maneira automatizada, permitindo a interação do engenheiro de requisitos para realizar os ajustes necessários ao modelo.

1.2 CONTEXTUALIZAÇÃO DA PESQUISA

Com o objetivo de fornecer uma sustentação para o desenvolvimento da pesquisa, um conjunto de conceitos relacionados com o assunto da pesquisa foi estudado. A Figura 1 apresenta a organização dos conhecimentos utilizados para o desenvolvimento desta dissertação.

A Engenharia de Software provê uma metodologia formalizada para o ciclo de vida de desenvolvimento do *software* e a Engenharia de Requisitos atua, justamente, nas fases iniciais do ciclo de vida, onde ocorre a especificação dos requisitos do *software* a ser desenvolvido. A ferramenta apresentada neste trabalho auxiliará na esfera da Engenharia de Requisitos, mas como essa disciplina está inserida na área de Engenharia de Software, devemos considerá-la no contexto geral da ferramenta desenvolvida neste projeto (vide Figura 1).

O artefato, modelo conceitual de classes, gerado pela ferramenta é um modelo que atende os padrões da UML. Esta linguagem é utilizada para especificar e documentar sistemas no paradigma orientado a objetos. O engenheiro

de requisitos deverá estar familiarizado com as notações da UML, principalmente no que diz respeito ao diagrama de classes, para que a ferramenta o auxilie a contento.

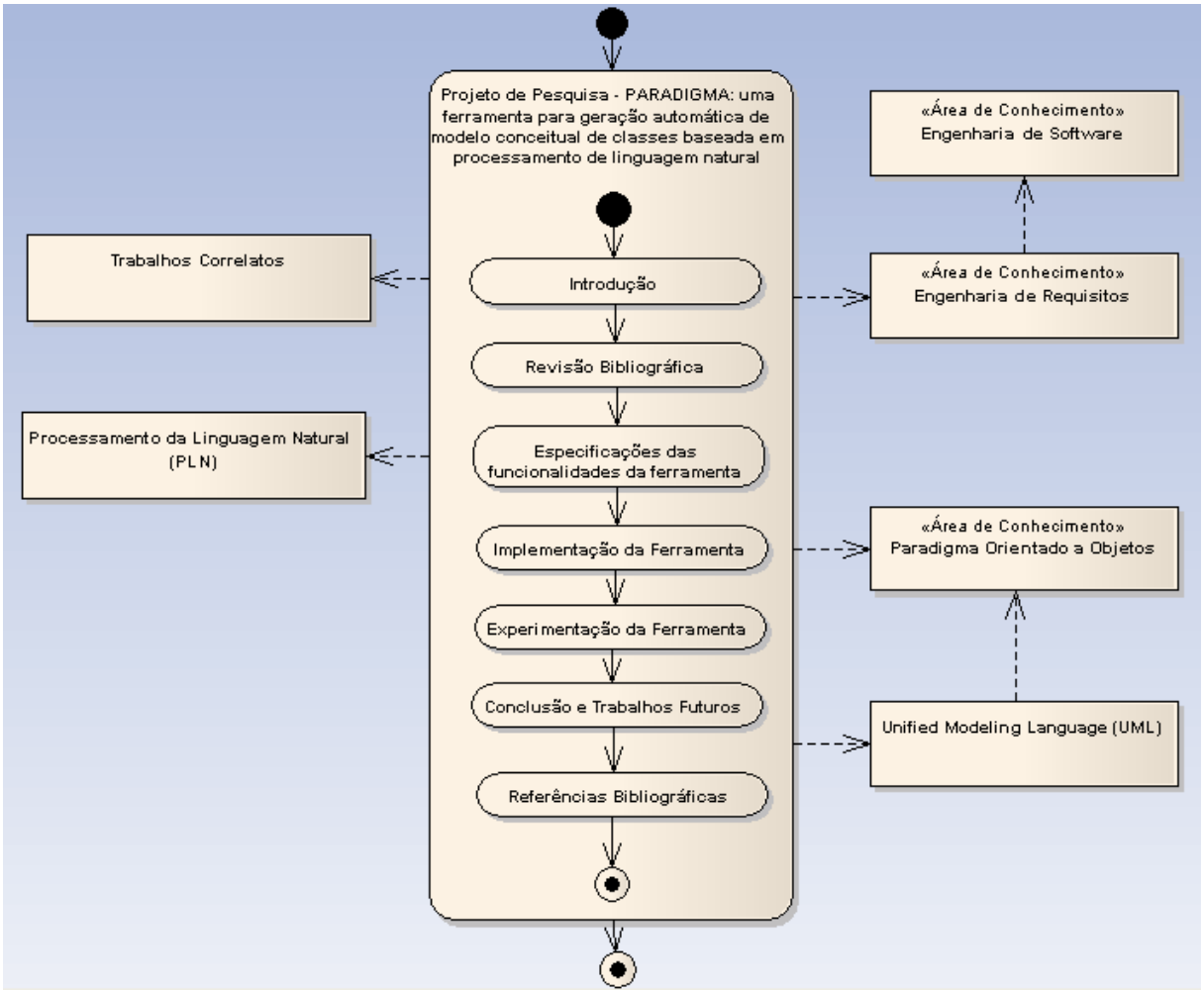


Figura 1: Contextualização da Pesquisa

Na Figura 1, é apresentada também a relevância do conhecimento referente ao processamento de linguagem natural. A ferramenta utiliza recursos provenientes do PLN para atingir o objetivo de receber como entrada descrições de requisitos funcionais em linguagem natural e gerar o modelo conceitual de classes a partir dessas descrições. De acordo com Aires (2000), a comunidade de PLN, no Brasil, vive um período de entusiasmo e resultados promissores. A razão disso é a possibilidade de contar com ferramentas de tratamento lingüísticos bastante abrangentes e independentes de Língua. A visão geral do processamento de linguagem natural e etiquetadores para Língua Portuguesa é apresentada no Capitulo 3.

Foram analisadas também várias ferramentas que se propõem a gerar artefatos no formato gráfico, a partir das descrições em linguagem natural para auxiliar os engenheiros de requisitos nessa fase inicial do ciclo de vida do software. Ficou evidente a necessidade de uma ferramenta nesse segmento que desse suporte à Língua Portuguesa e que permitisse a geração automatizada dos diagramas com a menor interferência possível do engenheiro de requisitos no processo.

1.3 ESTRUTURA DO TRABALHO

Os Capítulos 2 e 3 apresentam a revisão bibliográfica. No Capítulo 2 são abordados os conceitos de requisitos, principais atividades realizadas na Engenharia de Requisitos, as características de uma boa especificação de requisitos. No Capítulo 3 é apresentado o histórico do processamento da linguagem natural, os conhecimentos relevantes para o entendimento do PLN, etiquetadores de texto, estudos relacionados à utilização do PLN na Engenharia de Requisitos e as principais contribuições destas ferramentas.

O Capítulo 4 apresenta a visão geral da ferramenta, arquitetura da ferramenta, o etiquetador de texto elegido, os padrões lingüísticos pré-definidos, os diagramas utilizados no desenvolvimento, as interfaces e as funcionalidades da ferramenta PARADIGMA. Apresenta também o comparativo entre ferramentas que utilizam PLN no contexto da Engenharia de Requisitos.

O Capítulo 5 apresenta a experimentação da ferramenta PARADIGMA, mostrando os participantes, cenário de utilização da ferramenta, os resultados da experimentação e as análises dos resultados.

O Capítulo 6 apresenta a conclusão da dissertação e trabalhos futuros.

2 ENGENHARIA DE REQUISITOS

Quando o termo “crise do *software*” (NAUR; RANDELL, 1969) foi cunhado na década de 1960, um grande esforço foi direcionado para encontrar as causas do problema. A investigação concluiu que a deficiência na especificação de requisitos estava entre as causas mais importantes do problema. O desenvolvimento da especificação de requisitos, em muitos casos, parece trivial, mas é provavelmente onde ocorre a maioria das falhas (SCHWARTZ 1975).

De acordo com a complexidade e o tamanho do sistema a ser desenvolvido cresce a importância de uma boa especificação de requisitos. Desde a década de 1990, a Engenharia de Requisitos tem se estabelecido como uma subárea da Engenharia de Software (THAYER; DORFMAN, 1997). A Engenharia de Requisitos é uma área relativamente nova, surgindo oficialmente após a realização do Primeiro Simpósio Internacional de Engenharia de Requisitos na década de 1990.

Neste capítulo, são abordados os conceitos e as principais atividades da Engenharia de Requisitos para embasamento do projeto de pesquisa. Antes de definirmos o que é Engenharia de Requisitos, iremos definir primeiramente o que são requisitos.

2.1 REQUISITOS

Os requisitos são definidos, de forma mais geral, como algo que o sistema deverá realizar para atender as necessidades dos clientes e/ou usuários. Segundo Pfleeger (1998), um requisito é uma característica do sistema ou descrição de algo que o sistema é capaz de fazer para satisfazer o objetivo para o qual ele foi desenvolvido.

Leite (2001) define requisitos de software como sentenças, que expressam as necessidades dos clientes e que condicionam a qualidade do *sofware*. Robertson e Robertson (1999) definem requisitos como "alguma coisa que o produto tenha que fazer ou uma qualidade que necessita estar presente". Sob essa perspectiva, teríamos duas categorias de requisitos: aqueles responsáveis pela funcionalidade do sistema ("*... alguma coisa que o produto tenha que fazer...*") e aqueles responsáveis pelas qualidades que devem estar presentes, tais como,

desempenho, integridade, disponibilidade e segurança. Os primeiros são denominados requisitos funcionais e os últimos requisitos não-funcionais; essa classificação de requisitos será descrita a seguir. Sommerville (2001) apresenta os seguintes níveis de requisitos:

- **Requisitos de usuário:** são os requisitos abstratos de alto-nível; geralmente, estes requisitos são declarações em linguagem natural acompanhadas de diagramas, indicando quais serviços o sistema deverá prover e suas restrições.

- **Requisitos de sistema:** os requisitos do sistema têm uma descrição detalhada do que o sistema deve fazer, ou seja, é um conjunto dos serviços e restrições do sistema de forma detalhada. O documento gerado para detalhar os requisitos de sistema, geralmente chamado de especificação formal, deverá ser preciso, pois servirá de contrato entre a pessoa que está adquirindo o sistema e o desenvolvedor do *software*.

- **Requisitos de software:** é uma descrição do projeto do *software* que é a base para desenvolvimento mais detalhado e também da implementação. Esta especificação adiciona mais detalhes aos requisitos do sistema. Geralmente os requisitos de software são classificados como:

    - **Requisitos funcionais:** Descreve as funcionalidades ou serviços que são esperados que o sistema atenda. Estas funcionalidades são descritas em detalhes, incluindo as entradas, saídas e exceções do sistema. O conjunto de requisitos funcionais do sistema deverá ter as seguintes características: ser completo e consistente. Ser completo significa que todos os serviços requisitados pelos usuários devem ser definidos. Ser consistente significa que os requisitos não devem ser contraditórios.

    - **Requisitos não-funcionais:** são características do sistema, mas não funcionalidades ou serviços que tenham que ser implementados no sistema. Algumas características necessárias no sistema, que são classificadas como requisitos não-funcionais: desempenho, facilidade de uso, segurança, portabilidade, integridade.

- **Requisitos de domínio:** são requisitos derivados do domínio da aplicação do sistema, ao invés de uma necessidade específica dos usuários do sistema. Requisitos do domínio são importantes, porque geralmente eles refletem os fundamentos do domínio da aplicação. Se esses requisitos não forem satisfeitos, o sistema poderá não funcionar corretamente.

2.2 DEFINIÇÕES DE ENGENHARIA DE REQUISITOS

A Engenharia de Requisitos é a disciplina responsável por organizar e estruturar toda a base para construção do *software*. Mas o que vem a ser Engenharia de Requisitos? O uso do termo engenharia implica na utilização de métodos e técnicas que possibilitem a repetição do processo como um todo, permitindo que os requisitos sejam definidos de forma não-redundante, completos, corretos e de fácil entendimento por todos os envolvidos no processo (Usuários, engenheiros de requisitos, projetistas, enfim, todos os envolvidos no desenvolvimento).

De acordo com Sommerville (2001), a Engenharia de Requisitos é um processo que envolve todas as atividades necessárias para criar e manter os documentos de requisitos de sistema.

Lamsweerde (2000) define Engenharia de Requisitos como: “a identificação dos objetivos a serem atingidos pelo futuro sistema, a operacionalização de tais objetivos em serviços e restrições e a atribuição de responsabilidades pelos requisitos resultantes a agentes humanos, dispositivos e software”.

Robertson (2005) utiliza a seguinte definição: “Nós devemos ver a Engenharia de Requisitos como uma disciplina **sócio-técnica,** a qual requer uma diversidade de habilidades e conhecimentos”, ou seja, além dos engenheiros de requisitos focarem nas questões técnicas, eles devem também empregar recursos já utilizados por outras áreas de conhecimento, tais como, Psicologia, Sociologia e Lingüística.

Leite (2001) descreve Engenharia de Requisitos de uma forma mais completa,

> “Engenharia de Requisitos, uma subárea da Engenharia de Software, tem por objetivo tratar o processo de definição dos requisitos de software. Para isso, estabelece um processo no qual o que deve ser feito é elicitado, modelado e analisado. Este processo deve lidar com diferentes pontos de vista, e usar uma combinação de métodos, ferramentas e pessoal. O produto desse processo é um modelo, do qual um documento chamado requisitos é produzido. Esse processo é perene e acontece num contexto previamente definido a que chamamos de Universo de Informações. Universo de Informações é o contexto no qual o software deverá ser desenvolvido e operado. O UdI inclui todas as fontes de informação e todas as pessoas relacionadas ao software. Essas pessoas são também conhecidas como os atores desse universo. O UdI é a realidade circunstanciada pelo conjunto de objetivos definidos pelos que demandam o *software”*.

A seguir, serão abordadas as principais atividades desenvolvidas pela Engenharia de Requisitos.

2.3 PROCESSO DE ENGENHARIA DE REQUISITOS

Segundo Kotonya e Sommerville (1998), os processos de Engenharia de Requisitos variam radicalmente de uma organização para outra, ou até mesmo de um projeto para outro. Alguns fatores que levam a essa variação são: maturidade técnica, cultura organizacional, domínio da aplicação e o envolvimento disciplinado da equipe.

Como o processo de Engenharia de Requisitos pode variar em razão de alguns fatores, Kotonya e Sommerville (1998) afirmam que não existe um processo de Engenharia de Requisitos que possa ser considerado o “ideal” para todas as situações.

A escolha e adoção de um processo de Engenharia de Requisitos e de um conjunto de técnicas para elicitação de requisitos para um projeto, não é uma tarefa trivial. Um grande número de modelos de processos de Engenharia de

Requisitos está disponível. Cada um utiliza uma variedade de técnicas para auxiliar no processo de desenvolvimento do software. Foram identificados mais de vinte e seis modelos de processos de Engenharia de Requisitos e mais de cinqüenta e seis técnicas que podem ser utilizadas em conjunto com esses processos, dependendo do domínio da aplicação (JIANG et al., 2004).

Kotonya e Sommerville (1998) apresentam um modelo de processo de Engenharia de Requisitos e definem esse processo, como sendo, um conjunto organizado de atividades que levam à produção de um documento de especificação de requisitos.

Na Figura 2, esse processo é apresentado em uma visão de alto nível, dando ênfase nas entradas e saídas, não se preocupando em expor quais atividades e/ou passos foram executados para se chegar às saídas. Serão apresentadas, posteriormente, as atividades que compõem esse processo e as técnicas que auxiliam em cada atividade, como também, algumas dificuldades enfrentadas para atingir os objetivos.

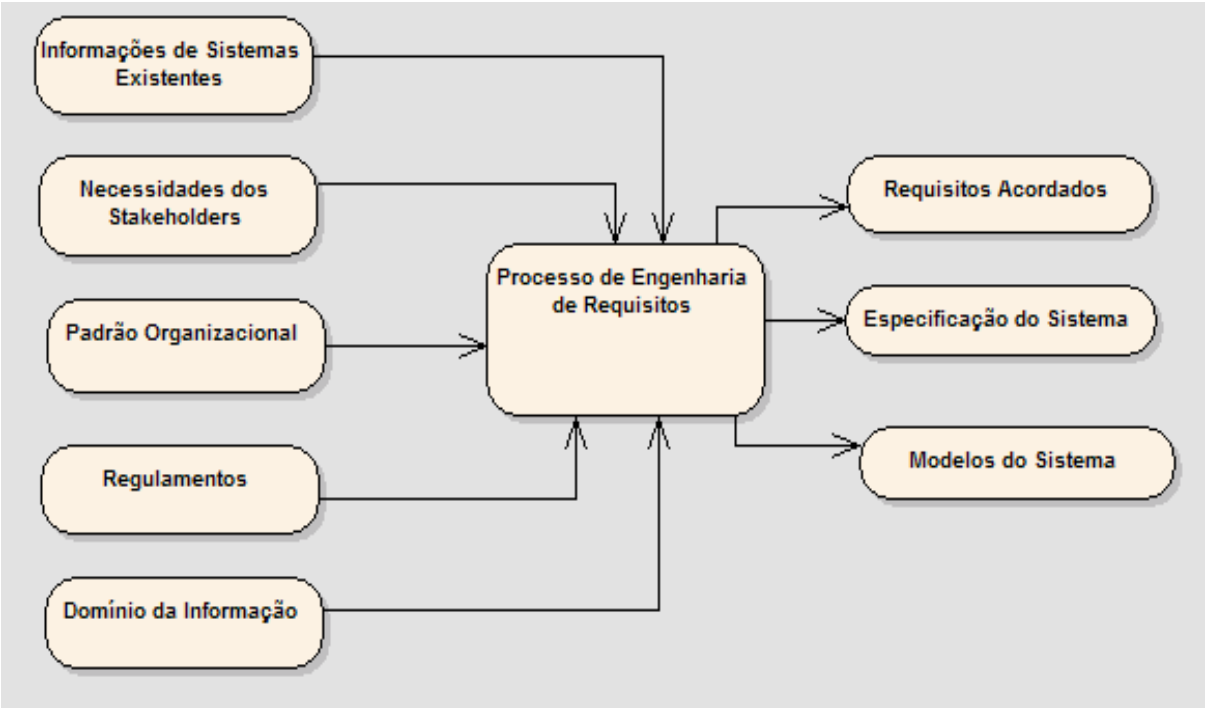


Figura 2 – Entradas e saídas do processo de Engenharia de Requisitos (KOTONYA; SOMMERVILLE, 1998)

Conjunto de entradas do processo de Engenharia de Requisitos apresentados na Figura 2:

- **Informações de sistemas existentes:** informações sobre as funcionalidades do sistema a ser substituído ou de sistemas que irão interagir com o sistema a ser especificado.

- **Necessidades dos *Stakeholders*:** descrição das necessidades dos *stakeholders* do sistema.

- **Padrão Organizacional:** padrões utilizados na organização referentes a padrões de desenvolvimento de sistema, gerenciamento de qualidade etc.

- **Regulamentos:** regulamentos externos que se aplicarão ao sistema.

- **Domínio da Informação:** informações gerais sobre o domínio da aplicação do sistema.

Conjunto de saídas do processo de Engenharia de Requisitos apresentados na Figura 2:

- **Requisitos Acordados:** descrição dos requisitos do sistema na forma em que os *stakeholders* entendam e concordem.

- **Especificação do Sistema:** descrição mais detalhada das funcionalidades do sistema.

- **Modelos do Sistema:** conjunto de modelos que descrevam o sistema sob diferentes perspectivas.

O processo de Engenharia de Requisitos proposto por Kotonya e Sommerville (1998) consiste em um conjunto estruturado de atividades para a especificação de requisitos do sistema/*software*; essas atividades são: elicitação, análise e negociação, especificação, validação e gerenciamento dos requisitos.

A Figura 3 mostra esse processo representado de forma espiral. As atividades no modelo espiral são executadas de forma repetitiva até se obter o documento de especificação de requisitos. Os requisitos que passam pelo ponto de decisão e são aceitos compõem o documento de requisitos. Os requisitos que são rejeitados por algum motivo, reiniciam a espiral, passando em cada atividade novamente, até que sejam aceitos ou totalmente descartados. Além das quatro

atividades apresentadas na Figura 3, existe a atividade de gerenciamento dos requisitos que ocorre em paralelo às demais atividades.

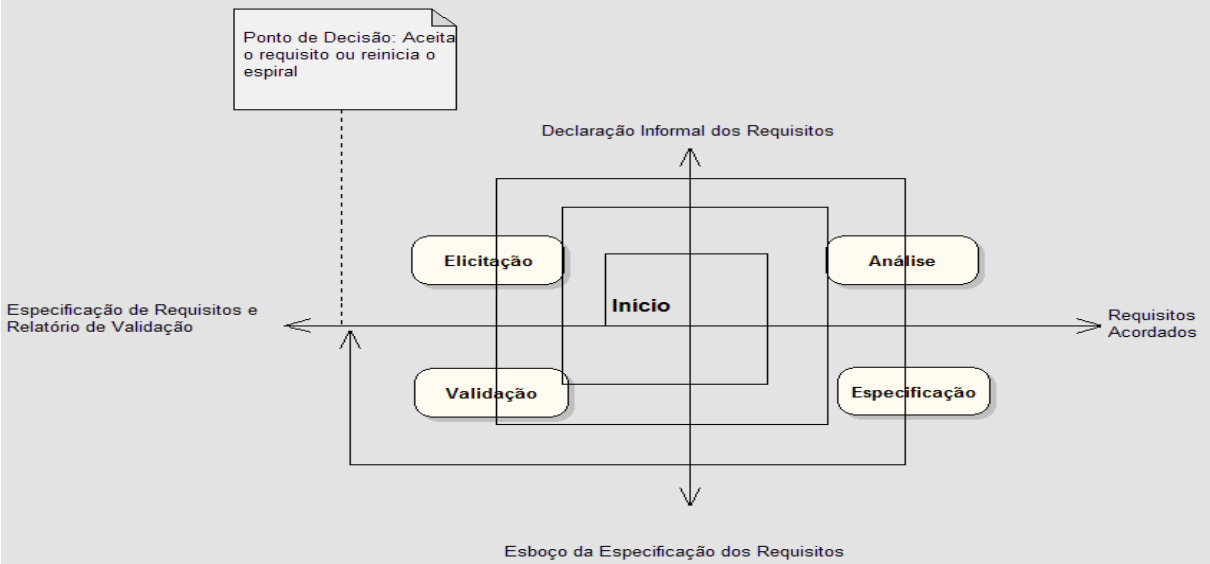


Figura 3 – Processo de Engenharia de Requisitos em espiral (KOTONYA; SOMMERVILLE, 1998)

As atividades que fazem parte do processo de Engenharia de Requisitos, proposto por Kotonya e Sommerville (1998), são descritas com mais detalhes a seguir.

2.3.1 ELICITAÇÃO DE REQUISITOS

A Atividade de elicitação de requisitos é muito importante dentro do contexto da Engenharia de Requisitos. Segundo Kotonya e Sommerville (1998), alguns componentes são primordiais na fase de elicitação. São eles:

- Conhecer o domínio da aplicação;
- Conhecer o problema a ser solucionado;
- Conhecer o contexto do negócio;
- Conhecer as Necessidades e Restrições dos *Stakeholders.*

Nessa fase, os engenheiros de requisitos em conjunto com os usuários, buscam a descoberta dos requisitos, tarefa que se funde também à fase de análise dos requisitos que será abordada na próxima subseção.

Na elicitação de requisitos é importante a interação do engenheiro de requisitos com os usuários do sistema. O entendimento do contexto onde o sistema funciona, a captura das reais necessidades dos usuários e a identificação dos problemas a serem solucionados é fundamental para, posteriormente, se especificar requisitos que atendam de forma satisfatória às necessidades dos usuários.

Quanto aos problemas enfrentados na fase de elicitação, podemos destacar dois grupos apontados por Brooks (1987), em um contexto mais amplo da Engenharia de Software, que são os problemas essenciais e os problemas acidentais. Martins (2001) contextualiza os problemas apontados por Brooks (1987) na visão da Engenharia de Requisitos. Esses problemas são descritos a seguir:

- **Problemas essenciais**

  - ✓ Dificuldade do usuário em saber efetivamente o que ele quer de um sistema computacional;
  - ✓ Dificuldade na comunicação entre quem deve prover a solução e os usuários;
  - ✓ Requisitos que sofrem alterações constantes, dificultando o gerenciamento de tais mudanças;
  - ✓ Tentativa de congelar os requisitos antes de começar a construção do software. Esta tarefa é difícil, pois os requisitos, em alguns casos, são dinâmicos.

- **Problemas Acidentais**

  - ✓ Documento de requisitos desenvolvido após o desenvolvimento do software (codificação);
  - ✓ Documento de requisitos focando várias visões diferentes, tornando-se confuso no seu objetivo;
  - ✓ Pouco esforço empregado para realizar a elicitação de requisitos, gerando um documento sem utilidade no ciclo de vida do software;

As dificuldades essenciais são partes intrínsecas do problema, já as dificuldades acidentais são resultados de falhas no controle intelectual para manter e gerir o que está sendo construído. Enquanto para as dificuldades essenciais não há uma fórmula mágica (“Bala de Prata”), para as dificuldades acidentais pode-se minimizar o problema através do processo de desenvolvimento sistemático, disciplinado e organizado.

2.3.1.1 TÉCNICAS PARA ELICITAÇÃO DE REQUISITOS

A questão básica em Engenharia de Requisitos é como descobrir exatamente o que os usuários necessitam para que o *software* desenvolvido atenda essas necessidades. A identificação dessas necessidades depende de uma série de fatores e entre esses fatores, podemos destacar os aspectos sociais, culturais e políticos.

Cada indivíduo envolvido no processo de elicitação de requisitos tem sua maneira de visualizar e transmitir conhecimentos. Para que possam ser atingidos todos os níveis de conhecimento, é necessária a utilização de técnicas diversificadas. Em cada projeto e dependendo do contexto, a escolha da técnica é essencial para o sucesso na elicitação de requisitos.

São abordadas algumas técnicas, selecionadas entre as mais conhecidas, que auxiliam o engenheiro de requisitos nessa etapa importante do ciclo de vida do software.

- **ENTREVISTA**

A entrevista é uma das técnicas de elicitação mais utilizada e é praticamente inevitável em qualquer desenvolvimento, já que é uma das formas de comunicação mais natural entre as pessoas (TORO; JIMENEZ, 2000).

Utilizando a entrevista, os engenheiros de requisitos entram em contato com vários *stakeholders,* discutem sobre o sistema em questão e constroem um entendimento sobre seus requisitos (KOTONYA; SOMMERVILLE, 1998).

De acordo com Kotonya e Sommerville (1998), as entrevistas são classificadas em 2 tipos:

- ✓ **Entrevista fechada:** o engenheiro de requisitos procura as respostas para um conjunto pré-definido de perguntas.

- ✓ **Entrevista Aberta:** o engenheiro de requisitos não tem um conjunto pré-definido de perguntas. Ele conversa com os *stakeholders* de maneira aberta procurando saber o que cada um espera do sistema que será desenvolvido.

As entrevistas realizadas de maneira improvisada podem não alcançar os objetivos desejados a partir de sua utilização. Piattini et al. (1996) *apud* (TORO e JIMENEZ (2000)) identificam 3 fases distintas que deve ter uma entrevista. São elas: preparação, realização e análise.

Na fase de preparação, o engenheiro de requisitos deverá realizar as seguintes tarefas: estudar o domínio do problema, selecionar as pessoas que deseja entrevistar, determinar o objetivo e o conteúdo (agenda) da entrevista, planejar o local, horário e tempo da entrevista.

A fase de realização é a ocorrência da entrevista propriamente dita. É nessa fase que o engenheiro de requisitos busca extrair os conhecimentos necessários sobre os requisitos do sistema. Nesse momento, o entrevistado deverá saber do que se trata a entrevista e o engenheiro de requisitos deverá conhecer o domínio do problema e os respectivos termos técnicos que poderão ser utilizados pelo entrevistado. Os horários devem ser seguidos conforme o cronograma e a condução da entrevista devem ter um formato bem definido onde apresente começo, desenvolvimento e encerramento da entrevista.

Na fase de análise da entrevista deverão ser lidas todas as notas tomadas durante a entrevista. O material deve ser organizado e estruturado. Nessa fase, outras fontes de informações, como por exemplo, entrevistas com outras pessoas e/ou documentos que descrevem os procedimentos do sistema a ser desenvolvido, poderão ser utilizadas para produzir o material final. O engenheiro de requisitos deverá encaminhar o resultado final da análise para ser validada pelo entrevistado.

- ***JAD – JOINT APPLICATION DEVELOPMENT***

JAD – Joint Application Development é uma técnica desenvolvida pela IBM, no final da década de 1970. É uma técnica orientada ao trabalho de grupo, onde os participantes trabalham como uma equipe compartilhando informações e

idéias sobre um determinado tema. Essa técnica é baseada em quatro princípios fundamentais:

- ✓ Dinâmica de grupo;
- ✓ Utilização de recursos visuais para melhorar a comunicação entre os participantes;
- ✓ Processo organizado e racional;
- ✓ Filosofia de documentação *WYSIWYG (What You See Is What You Get - O que você vê é o que você obtém).*

Em comparação com as entrevistas individuais, Toro e Jimenez (2000) destacam as seguintes vantagens do JAD:

- ✓ Ganho de tempo ao evitar que as opiniões dos clientes sejam analisadas em separado;
- ✓ Todo o grupo, incluindo os clientes e futuros usuários, revisa a documentação gerada, não somente os engenheiros de requisitos;
- ✓ Envolvimento maior dos clientes e usuários no projeto.

Os participantes de uma reunião JAD sempre se enquadram em um dos seguintes papéis: moderador, analista, executivo patrocinador, representantes dos usuários, representantes da área de tecnologia, especialistas. O papel do moderador é fundamental para a condução da reunião.

- **CASOS DE USO**

Os casos de uso podem ser utilizados para capturar o comportamento do sistema que está em desenvolvimento. É uma técnica que permite a interação entre os engenheiros de requisitos, usuários finais e especialistas de domínios para chegarem a uma compreensão comum das funcionalidades que devem ser implementadas no software (BOOCH et al., 2000).

Um caso de uso é a descrição de um conjunto de ações que o sistema deve executar em seqüência para chegar a um resultado esperado pelo ator. Os atores representam papéis que interagem com os casos de uso. Esses papéis podem ser exercidos por uma pessoa, por uma máquina ou por outro sistema que interage com o caso de uso.

Todos os casos de uso deverão representar algum comportamento distinto e identificável do sistema. Booch et al. (2000) dá algumas sugestões de como fazer um caso de uso bem-estruturado:

- Nomear um comportamento do sistema que deve ser único, identificável e razoavelmente atômico;
- Fatorar o caso de uso, obtendo o comportamento a partir do recurso de inclusão de outros casos de uso;
- Descrever o fluxo de eventos de maneira clara para facilitar o entendimento;
- Descrever um conjunto mínimo de cenários que especifiquem a semântica normal e a variante do caso de uso.

Visualmente um caso de uso é representado por uma elipse, o ator é representado por um boneco e a interação entre o ator e o caso de uso é representado por uma linha que os conectam. Para facilitar a comunicação entre os engenheiros de requisitos e os usuários são utilizados os diagramas de casos de uso. O diagrama de casos de uso representa o conjunto de casos de uso, os atores e seus relacionamentos. A Figura 4 apresenta um exemplo do diagrama de casos de uso.

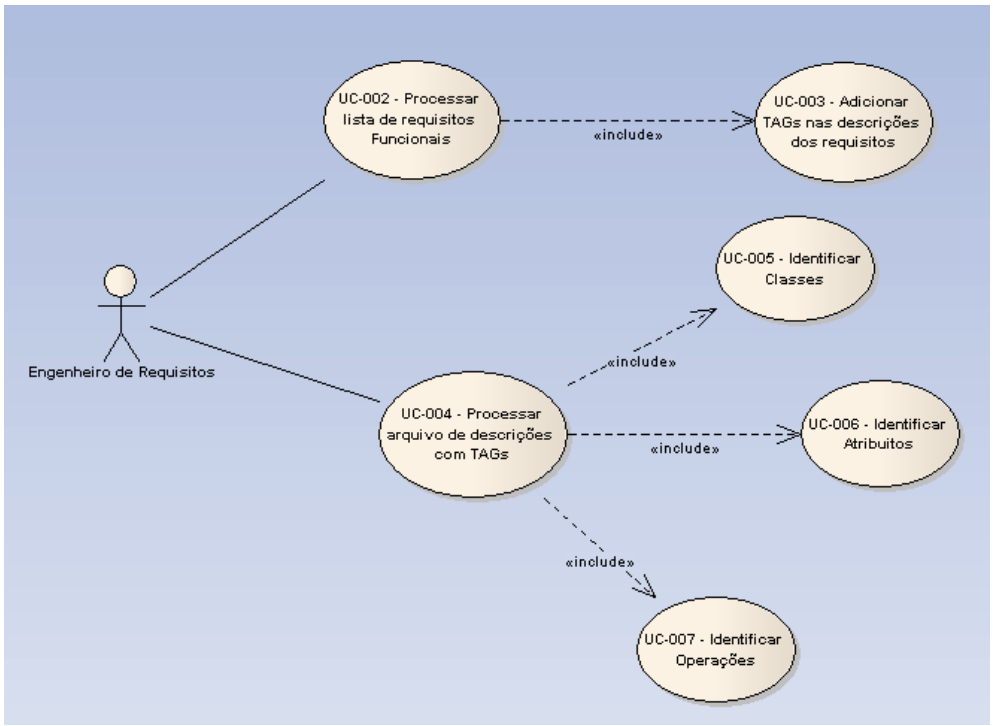


Figura 4 – Diagrama de Casos de Uso - Exemplo

- **PROTOTIPAÇÃO**

A técnica de prototipação consiste em construir um protótipo do sistema a ser implementado. A prototipação facilita a comunicação entre o engenheiro de requisitos e os usuários, por prover uma forma visual do sistema.

Para que a prototipação seja utilizada de forma eficaz, a construção do protótipo deve ser rápida. O que deve ficar evidente para os usuários, é que o protótipo não é o sistema real, mas um vislumbre do que se espera desenvolver efetivamente. O protótipo é um recurso interessante para definição de *interfaces* do sistema, elicitação de requisitos e validação de requisitos.

- **OBSERVAÇÃO (ETNOGRAFIA)**

Observação ou etnografia é uma técnica bastante utilizada pelas ciências sociais e vem sendo estudada pela Engenharia de Requisitos recentemente. Nessa técnica o engenheiro de requisitos se insere no cotidiano de trabalho dos usuários e adquire o conhecimento através da observação do trabalho feito por eles.

Alguns usuários têm dificuldades em descrever suas atividades, pois são triviais para eles, nesses casos, a melhor maneira de conseguir essas informações é observando o seu trabalho (KOTONYA; SOMMERVILLE, 1998).

2.3.2 ANÁLISE E NEGOCIAÇÃO DE REQUISITOS

Segundo Kotonya e Sommerville (1998), a atividade de análise do processo de Engenharia de Requisitos está diretamente vinculada a atividade de elicitação de requisitos. O objetivo da análise de requisitos é descobrir os problemas, as inconsistências e a não completitude dos requisitos elicitados. A análise de requisitos se intercala com a atividade de elicitação. Na medida em que os requisitos são elicitados, estes requisitos podem ser confrontados com uma lista de verificação de problemas, que serve de suporte para a análise. A lista sugerida por Kotonya e Sommerville (1998) inclui os seguintes itens:

- **Projeto prematuro:** os requisitos incluem informações voltadas a implementação?

- **Requisitos combinados:** a descrição do requisito descreve um único requisito ou pode ser desmembrado em vários requisitos?
- **Requisito desnecessário:** o requisito é fundamental? Ou é um requisito que não fará diferença se não for implementado?
- **Uso de software ou hardware:** a descrição do requisito deixa claro que deverá ser utilizado máquina ou software fora do padrão?
- **Conformidade com os objetivos do negócio:** o requisito é consistente com os objetivos do negócio definidos na introdução do documento de requisitos?
- **Ambigüidade do Requisito:** o requisito está descrito de forma ambígua? E pode ser lido de diferentes maneiras por diferentes pessoas? Quais são as possibilidades de interpretações dos requisitos?
- **Requisito realista:** o requisito poderá ser implementado dada a tecnologia disponível?
- **Requisito testável:** o requisito é passível de teste? Os engenheiros têm como montar um plano de teste, para validar se o software implementa o requisito descrito?

Divergências sobre os requisitos, quando existem vários *stakeholders,* é natural. Nesse ponto, surge a necessidade da atividade de negociação que está intercalada entre a atividade de elicitação e a atividade de análise. O objetivo da negociação de requisitos é discutir os conflitos entre requisitos e alcançar um ideal em que todos os *stakeholders* envolvidos concordem com a decisão tomada para a solução dos mesmos. Os conflitos não são falhas, mas refletem as diferentes prioridades e necessidades dos *stakeholders.*

2.3.3 ESPECIFICAÇÃO DE REQUISITOS

O artefato produzido nessa atividade é o documento de especificação de requisitos. De acordo com o documento IEEE Std 830-1998, a recomendação é que o documento de especificação de requisitos deverá constar de três seções principais:

- **Seção 1**: introduz o propósito e o escopo do documento de especificação de requisitos.

- **Seção 2**: descreve os fatores que afetam o sistema em questão e seus requisitos.

- **Seção 3**: descreve os requisitos do software.

Nessa fase, os requisitos elicitados e analisados serão transformados em documentos que organizam os requisitos dos sistemas. Por vários anos, os requisitos foram especificados utilizando somente linguagem natural, utilizando sentenças ou frases. Entretanto, ficou evidente que existem alguns problemas em se utilizar somente a linguagem natural para especificar os requisitos.

Primeiro, o problema da ambigüidade: para que o documento seja útil e sirva o seu propósito, todos os *stakeholders* envolvidos no processo de desenvolvimento do software deverão ter a mesma interpretação dos requisitos descritos no documento.

Segundo, a dificuldade de mostrar a rastreabilidade entre as características implementadas pelo software e os requisitos especificados em linguagem natural. Entretanto, engenheiros de software têm estudado várias maneiras de documentar as especificações de uma forma mais rigorosa e controlada, não utilizando somente linguagem natural, mas também, fazendo uso de técnicas de especificações semiformais e formais (PFLEEGER, 1998). Seguem abaixo alguns exemplos das técnicas semi-formais e formais utilizadas:

- **Técnicas semiformais**

  - ✓ DFD – Diagrama de Fluxo de Dados

  - ✓ DER – Diagrama de Entidade e Relacionamentos

  - ✓ UML – Unified Modeling Language

  - ✓ Outras

- **Técnicas formais**

  - ✓ Z

- ✓ VDM – Vienna Development Method
- ✓ CSP – Communicating Seqüencial Processes

O documento de especificação de requisitos oferece alguns benefícios, dentre os quais podemos destacar:

- Serve de veículo de comunicação entre os desenvolvedores e os usuários sobre o software a ser construído;
- Serve de base para estimativas de custos e cronograma;
- Serve de base para desenvolver o plano de testes;
- Serve como guia para ajudar os desenvolvedores a entenderem o que deve ser feito e quando terminaram o que deveria ser feito.

2.3.3.1 CARACTERÍSTICAS DE UMA BOA ESPECIFICAÇÃO DE REQUISITOS

De acordo com o documento IEEE Std 830-1998, a especificação de requisitos deve ser:

a) **Correta:** o requisito especificado no documento deverá ser implementado pelo software;

b) **Não Ambígua:** o requisito deverá ter somente uma interpretação;

c) **Completa:** Uma especificação é completa se incluir os seguintes elementos:

- Todo requisito significativo deverá ser conhecido e tratado;
- Definição das respostas do software para toda classe de entrada de dados e classes de situação;
- Rótulo e referências para todas as figuras, tabelas e diagramas na especificação de requisitos e definições de todos os termos e unidades de medidas;

d) **Consistente:** o requisito não pode ter conflito com nenhum outro requisito do sistema;

e) **Classificação por importância e/ou estabilidade:** uma especificação de requisitos é classificada por importância e/ou estabilidade, se cada requisito tem um identificador de importância ou estabilidade em particular;

f) **Verificável:** um requisito é verificável, se existir um processo finito para que uma máquina ou pessoa possa constatar a implementação de um requisito pelo software;

g) **Modificável:** é modificável se sua estrutura e estilo são de tal forma, que qualquer alteração possa ser feita facilmente, completamente e consistentemente, mantendo a estrutura e o estilo;

h) **Rastreável:** Uma especificação de requisitos de software é rastreável se a origem de cada requisito é clara e tem a facilidade de referenciá-lo no desenvolvimento futuro. Essa referência poderá ser feita através de um código seqüencial e único dos requisitos.

2.3.4 VALIDAÇÃO DE REQUISITOS

A validação de requisitos é a atividade que determina se o documento de especificação está consistente com as definições dos requisitos, ou seja, a validação assegura que os requisitos descritos irão atender realmente às necessidades dos usuários (PFLEEGER, 1998). Certifica se o documento de requisitos é uma descrição aceitável do sistema a ser implementado, verificando as seguintes características: completitude e consistência, padronização, conflitos de requisitos não detectados na fase de análise, erros técnicos e possíveis ambigüidades.

A validação de requisitos é feita utilizando o documento de especificação de requisitos na sua versão final, na qual os requisitos já foram elicitados, analisados e aceitos pelos *stakeholders* (KOTONYA; SOMMERVILLE, 1998). Os requisitos que não passarem pelo crivo da validação, deverão retornar para mais uma iteração no modelo espiral, até que seja validado.

A Figura 5 apresenta as entradas e saídas do processo de validação de requisitos.

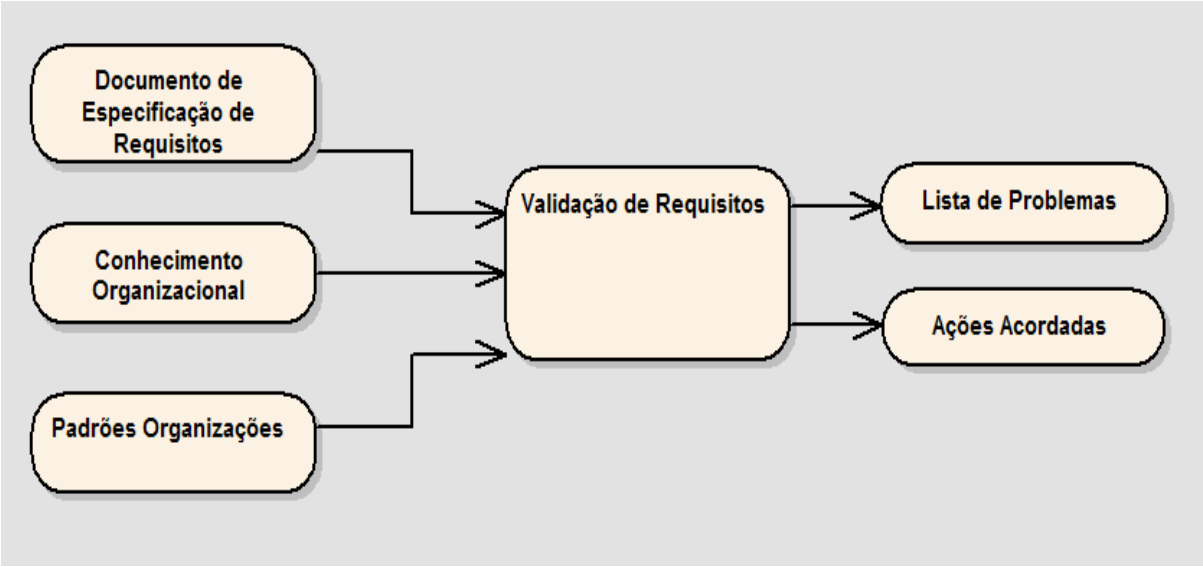


Figura 5 – Entradas e Saídas da Validação de Requisitos (KOTONYA; SOMMERVILLE, 1998)

Conjunto de entradas da atividade de validação de requisitos apresentados na Figura 5:

- **Documento de especificação de requisitos:** deve ser a versão completa do documento de especificação de requisitos, dentro dos padrões adotados pela organização.

- **Conhecimento organizacional:** fornece uma base para avaliar o grau de realismo dos requisitos do documento de especificação de requisitos.

- **Padrões:** padrões adotados pela organização para confecção dos documentos.

Conjunto de saídas da atividade de validação de requisitos apresentados na Figura 5:

- **Lista de Problemas:** lista de problemas encontrados no documento de especificação de requisitos.

- **Ações acordadas:** lista de ações acordadas para resolver os problemas encontrados.

2.3.5 GERENCIAMENTO DE REQUISITOS

De acordo com Kotonya e Sommerville (1998), o processo de gerenciamento dos requisitos ocorre durante todo o processo de Engenharia de Requisitos. Essa atividade dá suporte às demais atividades da Engenharia de Requisitos. As principais tarefas realizadas na atividade de gerenciamento são as seguintes:

- Gerenciamento de mudanças dos requisitos acordados;
- Gerenciamento dos relacionamentos entre os requisitos;
- Gerenciamento das dependências entre o documento de especificação de requisitos e os demais documentos gerados no processo de engenharia de sistema.

As mudanças nos requisitos ocorrem quando estes estão sendo elicitados, analisados, validados e até mesmo quando o sistema entra em produção. Alguns requisitos são mais suscetíveis às mudanças do que outros. Os requisitos estáveis geralmente são os que atendem a essência do sistema e seu domínio de aplicação, sofrendo mudanças lentamente. Os requisitos voláteis são os requisitos para atender um cliente ou um ambiente em particular, sendo mais suscetíveis a mudanças. Alguns fatores que levam a mudanças de requisitos:

- Erros, conflitos e inconsistências nos requisitos especificados;
- Evolução no conhecimento por parte dos usuários sobre o sistema em desenvolvimento;
- Problemas técnicos de custo ou cronograma;
- Mudanças nas prioridades dos clientes;
- Mudanças ambientais;
- Mudanças Organizacionais.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram abordados os conceitos de requisitos, as principais atividades realizadas na Engenharia de Requisitos, as características de uma boa especificação de requisitos e algumas técnicas utilizadas para auxiliar os engenheiros de requisitos na atividade de elicitação. No capítulo seguinte será apresentado um breve histórico do processamento de linguagem natural, os etiquetadores de texto, os conhecimentos relevantes para o entendimento da linguagem natural e os estudos relacionados com a pesquisa em questão, onde é descrito, de maneira sucinta, algumas ferramentas, no contexto da Engenharia de Requisitos, que fazem uso de PLN.

3 UTILIZAÇÃO DE PLN EM ENGENHARIA DE REQUISITOS

A motivação para estudos relacionados ao Processamento de Linguagem Natural é revolucionar a forma como o computador é utilizado. A maioria do conhecimento humano é registrada na forma de linguagem natural; computadores que puderem entender a linguagem natural poderão acessar toda essa informação de forma mais prática e rápida (ALLEN, 1995).

> "Hoje as pessoas de todos os segmentos da vida, incluindo profissionais, estudantes e a população em geral, são confrontadas pelo grande volume de informações, e esse grande volume é armazenado como texto não-estruturado. Em 2003 foi estimado que a produção anual de livros atingiria 8 Terabytes. (Um Terabyte é 1000 Gibabytes, o equivalente a 1000 caminhões lotados de livros.) Um ser humano levaria em média 5 anos para ler o novo material científico produzido em 24 horas. Todavia, essa estimativa é baseada no material impresso, devendo ser acrescido a informação disponível eletronicamente. ... Muitas pessoas têm gasto uma grande fração do tempo no trabalho ou lazer, navegando e acessando esse universo de informação na *Web".* (BIRD; KLEIN; LOPER, 2007).

Segundo Allen (1995), a linguagem é um dos aspectos fundamentais do comportamento humano e é um componente crucial em nossas vidas. Na forma escrita, ela serve de base de conhecimento que passa de uma geração para outra. Na forma falada, serve como principal meio de comunicação no dia-a-dia entre as pessoas.

A linguagem é a principal manifestação da inteligência humana. Através dela, expressamos as nossas necessidades básicas, aspirações, conhecimentos técnicos, fantasias e uma série de outras necessidades e emoções não listadas aqui. O grande desafio na Ciência da Computação tem sido construir máquinas inteligentes. A principal medida de máquina inteligente tem sido uma medida lingüística, chamada teste de *Turing*. O teste de *Turing,* de uma forma simplificada, parte da seguinte assertiva: o programa é inteligente se a pessoa que

participa do teste não for capaz de dizer se foi um programa ou um ser humano que respondeu as perguntas (BIRD; KLEIN; LOPER, 2007).

De acordo com Allen (1995), a linguagem é estudada por diferentes disciplinas e cada disciplina define seu próprio conjunto de problemas e como tratá-los. A Tabela 1 mostra as disciplinas e uma descrição rápida dos problemas típicos de cada disciplina.

Tabela 01 – Problemas típicos das disciplinas

Disciplinas	Problemas Típicos
Lingüística	Como as palavras formam frases e sentenças? (Estrutura da Linguagem)
Psicolingüística	Como as pessoas identificam a estrutura das sentenças? Como são identificados os significados das palavras?
Filosofia	O que é significado e como as palavras e sentenças o adquirem? Como as palavras identificam objetos no mundo real?
Lingüística Computacional	Como a estrutura das sentenças é identificada? Como o conhecimento e o raciocínio podem ser modelados? Como a língua pode ser usada para realizar tarefas específicas?

3.1 HISTÓRICO DO PROCESSAMENTO DE LINGUAGEM NATURAL

A percepção de que a linguagem natural poderia ser tratada de maneira computacional surgiu de um programa de pesquisa no início do século XX para reconstruir o raciocínio matemático utilizando a lógica. Aparece mais claramente nos trabalhos de Frege, Russell, Wittgenstein, Tarski, Lambek e Carnap (BIRD; KLEIN; LOPER, 2007). O Processamento de Linguagem Natural foi embasado em:

- Teoria da Linguagem Formal
  - ✓ Define uma linguagem como uma cadeia de caracteres aceita por uma classe de autômatos (autômatos finitos, linguagens livre de contexto, autômatos à pilha);
  - ✓ Provê suporte para a sintaxe computacional.

- Lógica Simbólica

  - ✓ Provê método formal para capturar aspectos relevantes da linguagem natural para expressar provas lógicas;

  - ✓ Um cálculo formal em lógica simbólica provê a sintaxe de linguagem. Cálculos com semântica e sintaxe bem definidas tornam possível associar significado às expressões de cálculo formal.

- Princípio da Composicionalidade

  - ✓ Percepção de que o significado de uma expressão complexa é composto dos significados de suas partes e composições;

  - ✓ Este princípio provê uma correspondência entre sintaxe e semântica, mostrando que uma expressão complexa pode ser computada recursivamente.

Nos anos 60, surgiram programas para compreensão da linguagem natural. Os computadores já eram capazes de aceitar e responder questões em inglês sobre assuntos variados, mas ainda de maneira rudimentar. Nessa década, também aconteceu a publicação da obra de Arthur Clark: *2001 Odisséia no espaço*, onde um dos personagens principais do filme é o computador inteligente HAL 9000. O Processamento da linguagem natural permite que os seres humanos possam interagir com os computadores de uma forma natural, fazendo uso da linguagem que eles estão acostumados. Elimina-se, desta maneira, a necessidade do aprendizado de uma linguagem artificial, onde a sintaxe é complicador nessa interação.

3.2 Conhecimentos Relevantes para Entendimento da Linguagem Natural

Segundo Allen (1995), as formas de conhecimento necessárias para o entendimento da Linguagem Natural são:

- Conhecimento fonético

  - – Identifica como as palavras estão relacionadas aos sons. Importante para sistemas baseados em diálogo.

- Conhecimento morfológico

  - Identifica como as palavras são construídas a partir da unidade de significado mais básico chamado **morfema**. Um morfema é a unidade de significado primitiva em uma linguagem. (Ex: Infelicidade resulta da combinação de 3 morfemas: in – **prefixo**, felic – **radical**, (i)dade – **sufixo**)

- Conhecimento sintático

  - Identifica como as palavras podem ser unidas para dar forma a sentenças corretas e determinam que papel estrutural cada palavra recebe na sentença e quais frases são subpartes de outras frases.

- Conhecimento semântico

  - Identifica o significado das palavras e como esses significados combinados formam sentenças com significado.

- Conhecimento pragmático

  - Identifica como as sentenças são utilizadas em diferentes situações e como essa utilização afeta sua interpretação.

- Conhecimento de mundo

  - Inclui o conhecimento geral de mundo que o usuário da linguagem deverá ter para manter uma conversa.

O conhecimento morfológico e conhecimento sintático são os conhecimentos explorados no desenvolvimento deste trabalho. A ferramenta desenvolvida não será capaz de interpretar a semântica de cada palavra dentro do texto, mas o enfoque será dado na classificação gramatical de cada palavra. A partir dessa classificação gramatical das palavras será possível realizar mapeamentos para os elementos do modelo conceitual de classes. Para obtenção da classificação gramatical de uma palavra dentro de um texto são utilizados etiquetadores de texto, que são apresentados a seguir.

3.3 ETIQUETADORES DE TEXTO

Com o avanço da tecnologia e dos computadores nos últimos anos, as informações estão disponíveis em grande quantidade no formato eletrônico. Essas informações estão dispostas nas mais variadas estruturas lingüísticas.

Segundo Aires (2000), a grande quantidade de textos, principalmente no formato eletrônico, fez com que os interesses por métodos empíricos de análise da Língua ressurgissem no início da década de 1990, fazendo com que os trabalhos na área de lingüística de *corpus* aumentassem significativamente.

Os etiquetadores (*taggers*) são as ferramentas utilizadas na etiquetagem automática de *corpus*. A etiquetagem automática tem sido bastante explorada pelo PLN. As etiquetas ou marcas, chamadas em inglês de *POS tags (Part-Of-Speech tags),* são, principalmente, classes gramaticais (morfossintáticas) das palavras do *corpus*.

Existem algumas abordagens para etiquetagem: abordagem lingüística, abordagem probabilística, abordagens híbridas. Essas abordagens não serão descritas neste trabalho; Aires (2000) expõe essas abordagens de uma forma simples e detalhada.

3.3.1 ETIQUETAGEM MORFOSSINTÁTICA DE TEXTOS

Para realizar a etiquetagem de um texto é necessária a definição de um conjunto finito de etiquetas (*tags*) e essas etiquetas devem ter um significado lingüístico associado. Geralmente o significado dessa etiqueta refere-se à categoria morfossintática ou gramatical de uma determinada palavra, dentro de uma língua e contexto.

O processo de etiquetagem consiste, então, em realizar a associação das palavras que aparecem em um dado texto a uma determinada etiqueta do conjunto finito de etiquetas. Essa associação é feita de acordo com o etiquetador utilizado.

A Figura 6 mostra as tarefas envolvidas no processo de etiquetagem automática. O processo é composto basicamente por três tarefas: escrutinador léxico, responsável por identificar os símbolos do texto (pontuação, palavras da língua, palavras estrangeiras, símbolos); classificador gramatical, responsável por

atribuir classes gramaticais às palavras; desambigüizador, responsável por resolver o problema da ambigüidade lexical, quando existem palavras diferentes com a mesma grafia e diferentes classificações gramaticais, o desambigüizador realiza a tarefa de analisar e atribuir somente uma classe gramatical para cada palavra do *corpus*.

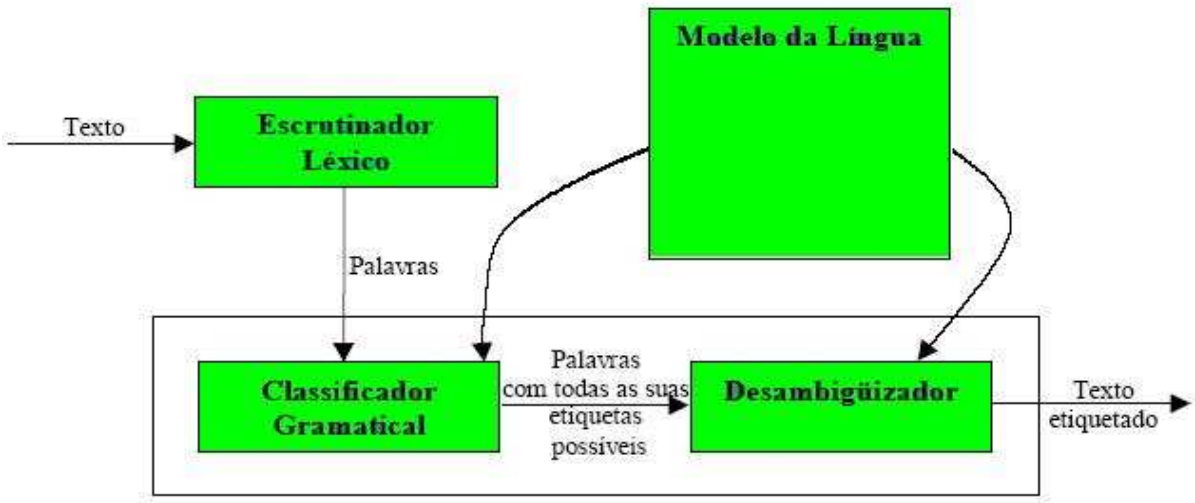


Figura 6: Processo de Etiquetagem Automática (AIRES, 2000)

De acordo com Aires (2000), na literatura de etiquetagem existem alguns critérios para avaliar a qualidade de um etiquetador. Esses critérios são:

- Sua precisão geral (*accuracy*), que é dada pelo número de palavras classificadas corretamente, dividido pelo número de palavras do seu arquivo de testes;
- Se o etiquetador é independente da Língua, ou seja, se pode ou não ser treinado para outras Línguas;
- Seu tempo de etiquetagem e tempo de treinamento;
- Qual o formato de entrada exigido;
- Se existem restrições quanto ao tamanho do conjunto de etiquetas, tanto para a precisão quanto para a complexidade do algoritmo de treinamento;
- Se existe a possibilidade de incrementar o léxico, isto é, fazer um treinamento incremental.

Na Tabela 2 são apresentados exemplos de etiquetas para a Língua Portuguesa.

Tabela 2 – Etiquetas para Língua Portuguesa

Adjetivo	ADJ
Advérbio	ADV
Artigo	ART
Numeral	NUME
Substantivo comum	N
Substantivo próprio	NP
Conjunção	CONJ
Pronome	PRON
Preposição	PREP
Verbo	VERB
Interjeição	I
Locução	LOCU
Palavra Denotativa	PDEN
Contração	PREP+ART PREP+PREP PREP+PD PREP+PPR PREP+PPOT PREP+ADJ PREP+N PREP+PPOA PREP+ADV PPOA+PPOA ADV+PPR ADV+PPOA ADJ+PPOA
Pontuação	. : ; - (! ? ...) " { } , '
Residuais: a etiqueta residual engloba: palavras estrangeiras, regionalismos, coloquialismos e símbolos especiais ($, %, @, #, etc..)	RES

3.4 Contribuições de Ferramentas que utilizam PLN na Engenharia de Requisitos

De acordo com MICH, FRANCH e NOVI INVERARDI (2003), ferramentas que utilizam processamento de linguagem natural na Engenharia de Requisitos podem trazer uma série de contribuições, dentre elas pode-se destacar:

- Auxiliar no gerenciamento de Requisitos;
- Permitir que os engenheiros de requisitos concentrem-se no problema, ao invés de se concentrarem na modelagem;
- Auxiliar na localização de ambigüidades e contradições nos documentos de especificação de requisitos;
- Extrair diretamente dos textos, em linguagem natural, elementos para gerar modelos conceituais;
- Auxiliar na documentação dos requisitos ;
- Rastreabilidade entre os textos em linguagem natural e os documentos produzidos (artefatos);
- Tradução dos documentos em várias línguas.

3.5 Trabalhos Correlatos

Vários estudos têm proposto recentemente a utilização de ferramentas lingüísticas para suporte à Engenharia de Requisitos (LI; DEWAR; POOLEY, 2004). Existem duas razões principais para que isso ocorra. Primeiro, o progresso feito na área de Processamento de Linguagem Natural. Segundo, a necessidade de prover aos desenvolvedores de software ferramentas que dêem suporte nos estágios iniciais do ciclo de vida do software (MICH; FRANCH; NOVI INVERARDI, 2003).

São apresentados nesta seção, de forma sucinta, alguns projetos que foram e estão sendo realizados na área de Engenharia de Requisitos com a utilização de processamento de linguagem natural.

3.5.1 PESQUISA DE MERCADO - FERRAMENTAS LINGÜÍSTICAS

Essa pesquisa foi realizada no verão de 1999 pelo departamento de ciência da computação e gerenciamento da Universidade de Trento. Esse estudo é parte de um projeto maior, cujo principal objetivo era mostrar as vantagens e desvantagens de pesquisas realizadas *on-line* em relação aos métodos tradicionais (MICH; FRANCH; NOVI INVERARDI, 2003). A pesquisa realizada pretendia:

- Avaliar as vantagens econômicas de desenvolver uma ferramenta CASE (Computer Aided Software Engineering) que integrasse técnicas lingüísticas de análise para documentos escritos em linguagem natural;

- Verificar a existência de demanda em potencial para tais ferramentas.

O resultado da pesquisa constatou que a maioria dos documentos disponíveis para o processo de Engenharia de Requisitos é provida pelos *stakeholders,* e estão descritos em linguagem natural, levando à conclusão que o uso de ferramentas e técnicas lingüísticas pode exercer um papel crucial no suporte à análise dos requisitos.

Para a utilização das ferramentas CASE baseadas em PLN, dois nichos distintos de mercado foram identificados. O primeiro é composto pelas companhias que já adotaram um método de Engenharia de Software para o desenvolvimento do software; neste caso, a ferramenta CASE baseada em PLN seria anexada à ferramenta já adotada pela companhia, a qual daria suporte para as demais fases do desenvolvimento. O segundo nicho de mercado é composto pelas que adotariam a ferramenta CASE baseado em PLN para facilitar a adoção de uma metodologia, ou melhores práticas no desenvolvimento de software, pois vêem o processo de Engenharia de Requisitos como um processo crucial no desenvolvimento de sistemas.

Algumas observações referentes às características que os engenheiros de requisitos de pequenas e médias empresas esperam de uma ferramenta baseada em PLN de acordo com a pesquisa:

- A possibilidade de acelerar a produção dos modelos de análise e rapidamente criar modelos que sirvam de interação entre os usuários e os grupos do projeto;

- A ferramenta poderia ser utilizada para treinamento de engenheiros de requisitos iniciantes e/ou para treinamento de engenheiros de requisitos que não tenham familiaridade com o paradigma Orientado a Objetos (Ferramenta deveria atender este paradigma);

- Possibilidade de integração da ferramenta baseada em PLN com ferramentas CASE do mercado.

3.5.2 Análise de Requisitos em Linguagem Natural e Geração de Modelo de Classe - UCDA

A ferramenta UCDA utiliza uma metodologia para automatizar a análise de requisitos em linguagem natural e a geração de modelos de classes baseado no RUP *(Rational Unified Process).* A utilização de Casos de Uso é proposta para reduzir a complexidade e a liberdade permitida pela linguagem natural.

Algumas regras são identificadas e utilizadas para automatizar a geração de diagramas de classes a partir das especificações dos casos de uso. A ferramenta CASE UCDA (Use-Case driven Development Assistant) foi desenvolvida para dar suporte à metodologia descrita acima. A ferramenta UCDA pode auxiliar o desenvolvedor na geração de diagramas de casos de uso, especificações de casos de uso, diagramas de colaboração e diagramas de classe no IBM Rational Rose, ajudando assim a acelerar a modelagem do diagrama de classes e a análise de requisitos, reduzindo o tempo de desenvolvimento do software (Liu et al., 2004).

3.5.3 CM-Builder: uma Ferramenta CASE PLN

O *Class Model Builder* (CM-Builder) é uma ferramenta CASE modular baseada em linguagem natural que realiza análise OO independente do domínio da aplicação (Harmain; Gaizauskas, 2000). A ferramenta recebe um documento de especificação de requisitos como entrada, analisa esta entrada linguisticamente e extrai desta análise as principais classes de objetos e os relacionamentos estáticos entre os objetos dessas classes. A ferramenta produz três tipos de saída:

- Uma lista de palavras que são pré-classificadas como classes;

- Uma lista de palavras que são pré-classificadas como relacionamentos;
- Um modelo conceitual.

Os passos executados pela ferramenta para gerar as saídas especificadas acima, com base no documento de especificação de requisitos, são:

1. Obter um conjunto de requisitos funcionais ou a descrição do problema em linguagem natural;
2. Utilizar um sistema de Processamento de Linguagem Natural para analisar semanticamente e sintaticamente o texto informal de requisitos e guardar o resultado intermediário da análise para uma análise mais detalhada posteriormente;
3. Utilizar os resultados produzidos pelo sistema de Processamento de Linguagem Natural para extrair as classes, atributos e os relacionamentos;
4. Gerar um modelo gráfico, permitindo a interação do analista. Nesse momento, são feitas algumas correções no modelo gerado pela ferramenta.

3.5.4 MODELAGEM CONCEITUAL ATRAVÉS DA ANÁLISE LINGÜÍSTICA - LIDA

Existe um grande número de métodos para especificação e desenvolvimento de requisitos, mas poucas ferramentas para auxiliar os engenheiros de requisitos na transição das descrições textuais (linguagem natural) para outras notações (UML ou outros modelos conceituais) (Overmyer; Lavoie; Rambow, 2001).

Apesar das vantagens que a tecnologia orientada a objetos pode prover para a comunidade de desenvolvimento de software e seus clientes, os problemas fundamentais associados à identificação dos objetos, seus atributos e métodos ainda permanecem. Esses processos ainda são manuais, na maioria das vezes, e dependem da experiência dos engenheiros de requisitos. A ferramenta LIDA (*Linguistic Assistant for Domain Analysis*) é uma ferramenta desenvolvida para

auxiliar na modelagem conceitual dentro do paradigma Orientado a Objetos e supre essa necessidade de ferramentas nas fases iniciais do ciclo de vida do software (Overmyer; Lavoie; Rambow, 2001).

A ferramenta LIDA tem as seguintes características:

- Processamento lingüístico independente do domínio do problema (identifica substantivo, verbo, adjetivo);
- Texto completo, lista de palavras, e modelos UML são mostrados em paralelo, para que o analista compare as diferentes visões;
- As palavras podem ser associadas a tipos do modelo (classes, atributos, papéis etc.) com um simples clique. A palavra é pintada no texto e será mostrada graficamente na janela do diagrama;
- As palavras podem ser ordenadas alfabeticamente, por freqüência ou por tipo associado;
- Permite visualizar as palavras-chave em um contexto;
- Descrições do modelo poderão ser obtidas através de *links*;
- Modelos completos podem ser exportados para ferramentas CASE para refinamentos e também poderão ser importados para a ferramenta LIDA para validação dos textos.

A metodologia utilizada na ferramenta LIDA envolve uma identificação inicial dos elementos do modelo através da análise assistida do texto, seguido pelo refinamento através das validações feitas pelo engenheiro de requisitos, usando descrições textuais do modelo a ser desenvolvido. A ferramenta LIDA oferece uma boa integração entre o processo de análise do texto e o processo de modelagem conceitual, mas a análise do texto permanece em boa parte no processo manual. A ferramenta LIDA não provê um fim, mas um começo no processo de análise de requisitos. Os modelos gerados a partir desta ferramenta, serão utilizados como ponto de partida na fase de projeto no ciclo de vida do software.

3.6 CONSIDERAÇÕES FINAIS

O processamento de linguagem natural foi importante para o desenvolvimento da ferramenta proposta. Utilizaram-se recursos desenvolvidos por grupos de pesquisa da área de lingüística, principalmente os etiquetadores de textos, eles servirão de ponte entre os textos em linguagem natural e a ferramenta desenvolvida neste projeto.

Apresentou-se neste capítulo, além dos conceitos envolvendo o PLN, algumas ferramentas da área de Engenharia de Requisitos que fazem uso do PLN e constatou-se a real necessidade de ferramentas que ofereçam suporte ao engenheiro de requisitos nessa fase importante do ciclo de vida do *software* e que dêem suporte para a Língua Portuguesa.

No próximo capítulo, será apresentada a ferramenta PARADIGMA, destacando suas funcionalidades, sua macro-arquitetura e a linguagem utilizada para desenvolvê-la. Serão apresentados também os principais modelos utilizados no processo de desenvolvimento da ferramenta: o diagrama de casos de uso, o diagrama de classes, o diagrama entidade relacionamento, o diagrama de atividades e, por fim, realizaremos um comparativo entre as ferramentas desenvolvidas em outros projetos de pesquisa, que visam auxiliar os engenheiros de requisitos nas atividades de elicitação e especificação de requisitos e a ferramenta PARADIGMA.

4 PARADIGMA: UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE MODELO CONCEITUAL DE CLASSES A PARTIR DA ESPECIFICAÇÃO DE REQUISITOS EM LINGUAGEM NATURAL

4.1 VISÃO GERAL DA FERRAMENTA

Nesta seção apresentamos a visão geral da ferramenta PARADIGMA, desenvolvida para auxiliar o engenheiro de requisitos na fase de concepção de um projeto de *software*.

A maioria das especificações de requisitos de software é escrita em linguagem natural, e a partir destes documentos de especificação, são gerados os demais documentos necessários para o desenvolvimento do software. A ferramenta, proposta e desenvolvida neste trabalho, utiliza recursos do PLN em conjunto com padrões lingüísticos identificados nos textos de especificação de requisitos, permitindo a geração automatizada de modelos conceituais de classes utilizando a notação da UML.

Na Figura 7 é apresentado um diagrama de atividades contendo as atividades da ferramenta PARADIGMA. A seguir, é descrita cada atividade identificada no diagrama.

O modelo conceitual de classes poderá ser obtido através do processo manual, automatizado ou híbrido. No processo manual, o engenheiro de requisitos deverá desenhar o modelo de classes da forma tradicional, identificando as classes, atributos, operações e associações através do método heurístico.

No processo automatizado, a ferramenta recebe como entrada, o texto em linguagem natural. Esse texto pode ser um conjunto de requisitos especificados, cenários de casos de uso ou documentos resultantes das técnicas utilizadas no processo de elicitação de requisitos. A utilização da linguagem natural traz consigo o problema da ambigüidade. De acordo com Daniel Berry e Erik Kamsties (2004), a ambigüidade é um fenômeno do mundo real que é estudado por uma gama de disciplinas, tais como: Lingüística, Filosofia, Direito, Psicolingüística e não poderia deixar de mencionar Engenharia de Software. Daniel Berry e Erik Kamsties (2004) declaram que não existe uma especificação sem ambigüidade. O

que existe é uma especificação eficaz. Uma especificação eficaz é uma especificação que é entendida pela maioria das pessoas envolvidas no processo de desenvolvimento do software, permitindo que o software realize o que a maioria espera que ele o faça.

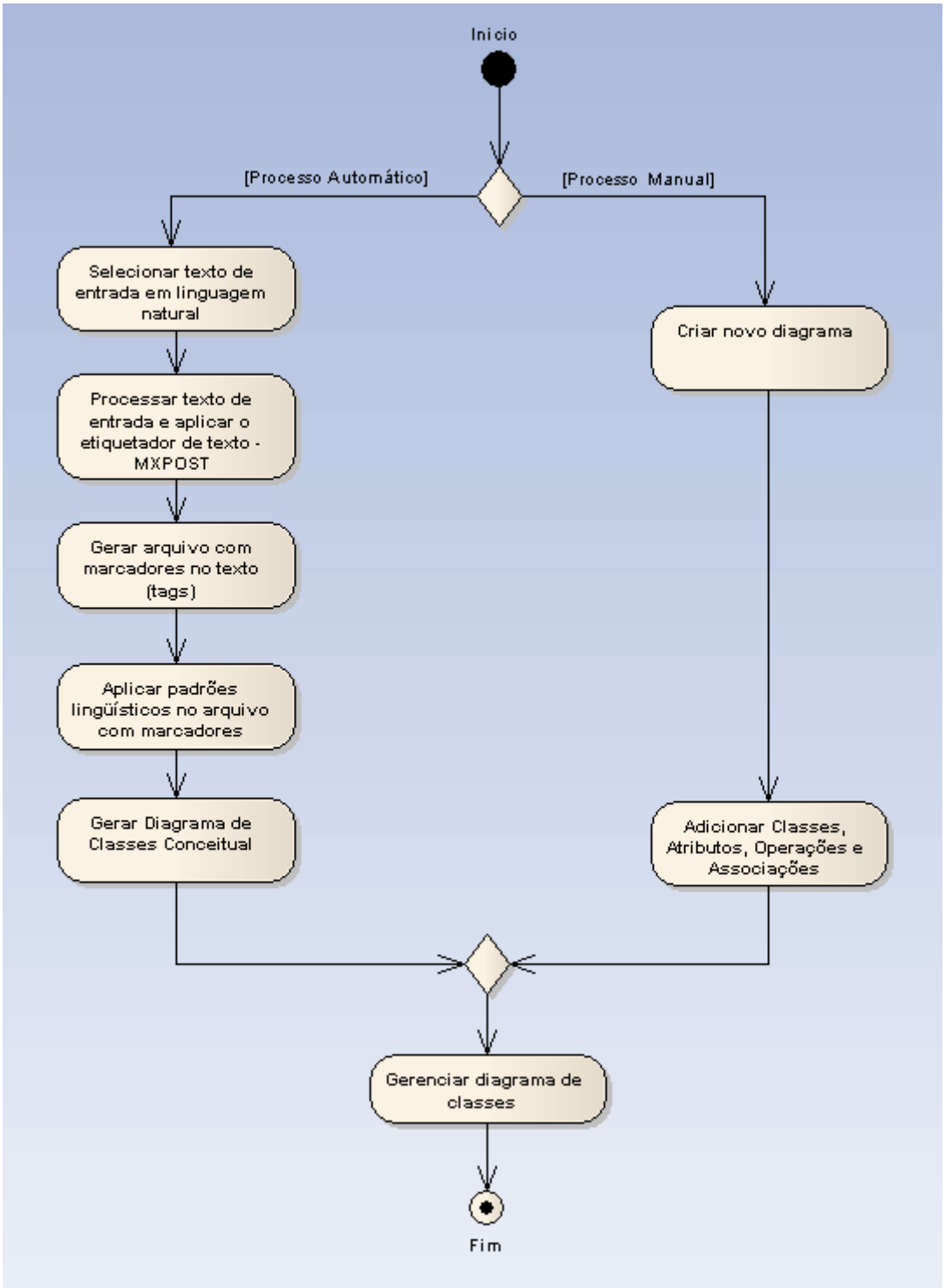


Figura 7 – Diagrama de Atividades da ferramenta PARADIGMA

O texto de entrada é formatado no padrão exigido pelo etiquetador de texto MXPOST (seção 4.1.2). O etiquetador recebe o texto formatado como entrada e gera um arquivo contendo as marcações das palavras. Essas marcações identificam as palavras em categorias morfossintáticas.

São aplicados os padrões lingüísticos pré-definidos (seção 4.1.3) no arquivo contendo as marcações e, através desses padrões, cada palavra é classificada nos tipos: classe, atributo e operação (método). As associações entre as classes e respectivas multiplicidades são identificadas de acordo com a relação existente entre as palavras candidatas a classe no texto de entrada. Após a identificação de todos os elementos que compõem o modelo de classes, ele é gerado de forma automatizada. O engenheiro de requisitos deverá interagir no processo automatizado para vincular os atributos e operações às classes respectivas.

No processo híbrido, ocorre a geração automatizada do modelo de classes, conforme descrito acima, e o engenheiro de requisitos interage com o modelo para realizar os devidos ajustes.

Independentemente do processo utilizado para chegar ao modelo conceitual de classes, o engenheiro de requisitos poderá interagir com o modelo fazendo as alterações necessárias. O engenheiro de requisitos poderá realizar as seguintes tarefas:

- Adicionar classes, atributos, operações e associações que não foram identificadas durante o processo utilizado para gerar o modelo conceitual de classes (processo automático, manual ou híbrido);
- Excluir classes, atributos, operações e associações indevidas;
- Vincular os atributos e operações às respectivas classes;
- Modificar as multiplicidades das associações;
- Distribuir os elementos no diagrama de forma organizada, utilizando o recurso de arrastar e soltar do Windows;
- Adicionar, de forma automatizada, mais classes ao modelo, ou seja, informar outro arquivo com texto em linguagem natural para adição de classes em um diagrama existente;
- Salvar os modelos de classes;
- Excluir modelos de classes;

- Abrir modelos de classes;
- Alterar uma associação simples para uma generalização (a ferramenta não identifica as generalizações de forma automática).

Toda essa interação do engenheiro de requisitos com o modelo foi sintetizada na atividade: gerenciar modelo de classes.

O desenvolvimento da ferramenta PARADIGMA foi realizado utilizando os seguintes recursos:

- **Linguagem de Programação:** C++ (ambiente da *Borland Developer Studio* 2006). A escolha da linguagem de programação foi motivada pela necessidade dos recursos de orientação a objetos e pelo que a linguagem C++ representa no meio acadêmico. Os arquivos fontes da ferramenta PARADIGMA serão disponibilizados para a continuidade de pesquisas nesse contexto.
- **Banco de dados:** Microsoft Office Access 2003. A utilização do banco de dados access restringe a utilização da ferramenta ao sistema operacional Windows; na próxima versão da ferramenta o banco de dados deverá ser alterado para um que possibilite uma maior portabilidade. Como se trata de uma versão desenvolvida no contexto acadêmico, a utilização do banco de dados Access atendeu às expectativas;
- **Ferramenta de Modelagem:** Enterprise Architect da Sparx Systems, Program Version: 7.0.818.

4.1.1 ARQUITETURA DA FERRAMENTA

A arquitetura da ferramenta PARADIGMA é apresentada na Figura 8. Nessa visão de alto nível, são apresentados os módulos principais da ferramenta, que englobam os textos em linguagem natural, a etiquetagem desse texto, a aplicação dos padrões lingüísticos no arquivo contendo as palavras com as marcações, a interação do engenheiro de requisitos no ambiente visual e a geração do modelo conceitual de classes como artefato de saída.

Os textos de entrada devem estar no padrão ASCII, para que o etiquetador de texto possa gerar um arquivo similar com as marcações necessárias nas palavras. Os textos em linguagem natural devem ser ajustados no formato

exigido pelo etiquetador. Esse formato requer que cada palavra, símbolo e pontuação do texto estejam separados por um espaço em branco.

O etiquetador de texto realiza a etiquetagem morfossintática no texto de entrada já formatado. Essa etiquetagem é feita com base na definição de um conjunto finito de etiquetas (*tags*) e essas etiquetas devem ter significados lingüísticos associados a elas. O artefato de saída do etiquetador é um arquivo no padrão ASCII contendo o texto de entrada devidamente etiquetado.

O etiquetador utilizado na ferramenta PARADIGMA é o MXPOST. Algumas informações importantes sobre este componente e a justificativa para sua utilização estão descritas na seção 4.1.2.

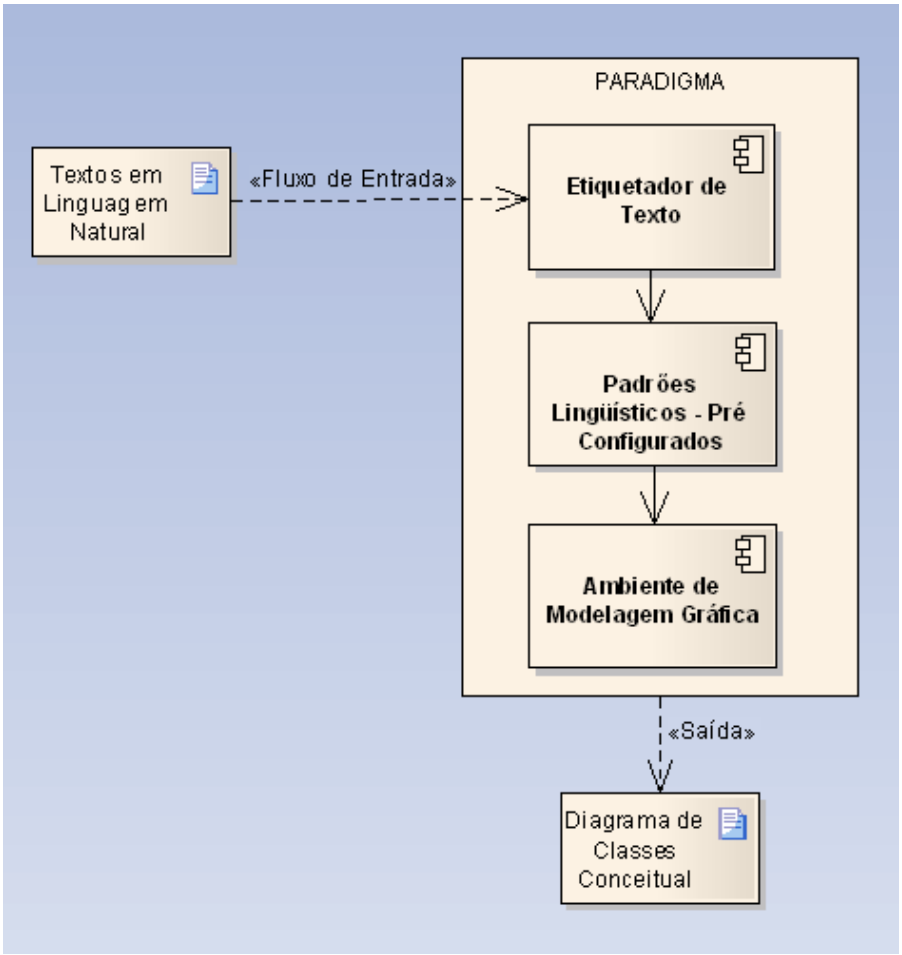


Figura 8 – Macro-arquitetura da ferramenta PARADIGMA

Os padrões lingüísticos são o diferencial da ferramenta PARADIGMA; através deles é possível aproximar os modelos de classes gerados de maneira automatizada, dos modelos criados por modelador humano. Na seção 4.1.3 serão apresentados os padrões lingüísticos pré-definidos na ferramenta e sua ordem de prioridade.

O ambiente de modelagem gráfica, ou *interface* gráfica, permite a interação do engenheiro de requisitos de maneira amigável e intuitiva com o modelo de classes gerado pela ferramenta, sendo possíveis os ajustes necessários ao modelo. Como artefato de saída temos o modelo conceitual de classes gerado a partir dos textos em linguagem natural e ajustados pelo engenheiro de requisitos.

4.1.2 ETIQUETADOR DE TEXTO – MXPOST

Aires (2000) realizou um estudo comparativo que reuniu pelo menos um etiquetador por abordagem: um estatístico, um neural e um híbrido. Os etiquetadores escolhidos foram os que estavam disponíveis na *web* e que tivessem sido utilizados em vários experimentos em outras línguas e/ou apresentassem a maior precisão geral para o inglês.

Os etiquetadores escolhidos para o estudo comparativo foram: *TreeTagger, TBL, MXPOST e PoSiTagger.* Esses etiquetadores tinham as características desejadas para o estudo em questão.

A avaliação dos etiquetadores foi feita baseada nas seguintes metas:

- Verificar o tempo gasto em cada etiquetador com o treinamento e etiquetagem;
- Calcular a precisão geral de cada etiquetador;
- Utilizar o algoritmo e *Bootstraping* para estimar a taxa de erro do etiquetador de treinamento mais rápido;
- Verificar as etiquetas mais problemáticas para cada etiquetador;
- Averiguar se existia um etiquetador que apresentasse uma melhor precisão com o corpus e conjunto de etiquetas propostos;
- Averiguar a existência de um conjunto de etiquetas problemáticas para todos os etiquetadores;
- Checar se haveria um acréscimo de precisão no etiquetador de maior precisão com o aumento do *corpus* de treinamento;
- Verificar a importância das palavras desconhecidas. Para fazer essa verificação, seriam feitos testes com *corpus* de treinamento, onde 100% das palavras fossem conhecidas, verificando assim a ocorrência de acréscimo de precisão ou não do etiquetador.

De acordo com Aires (2000), o MXPOST foi o etiquetador que apresentou o maior índice de precisão geral, atingindo o índice de 89,66%. A precisão geral depende muito do *corpus* de treinamento existente, do seu tamanho e uma taxa de erro de etiquetagem manual mínima.

Para a Língua Inglesa, os etiquetadores já atingiram o estado da arte, com um nível de precisão geral próximo de 100%, mas para a Língua Portuguesa ainda faltam *corpus* treinados com a robustez necessária.

O MXPOST é um etiquetador baseado do modelo de máxima entropia e foi desenvolvido por Ratnaparkhi (1996). Ele foi escolhido para integrar a ferramenta PARADIGMA pelas seguintes razões:

- Foi utilizado em projetos similares ao que propomos, em outra Língua (Língua Inglesa);
- É um etiquetador independe de língua, ou seja, podem ser incorporados *corpus* e treinados em línguas variadas, inclusive para o Português;
- Obteve um índice de precisão geral superior aos demais etiquetadores avaliados por Aires (2000);
- Estava disponível na *web* para utilização em projetos de pesquisa;
- Facilidade de integração com a ferramenta PARADIGMA;
- Portabilidade, pois foi desenvolvido em JAVA.

4.1.3 PADRÕES LINGÜÍSTICOS PRÉ-DEFINIDOS

A ferramenta PARADIGMA não é capaz de realizar a análise semântica das frases, para suprir essa tarefa foram definidos padrões lingüísticos. Os padrões lingüísticos, nada mais são do que os mapeamentos dos elementos, ou seja, palavras dentro de uma frase. Esse mapeamento tomará como base a classificação gramatical (classificação sintática) de cada palavra e seu posicionamento dentro da frase. De acordo com os padrões lingüísticos, define-se que tratamento será dado àquela palavra. Se ela é candidata a ser uma classe, atributo, operação ou se deveremos simplesmente desprezá-la.

A proposta original era permitir que o engenheiro de requisitos pudesse configurar seus próprios padrões lingüísticos, permitindo que a ferramenta se adequasse à forma de escrita de cada indivíduo. Com as definições de alguns

padrões lingüísticos básicos, percebemos que uma série de fatores influencia no resultado final, tais como:

- Prioridade de aplicação dos padrões lingüísticos;
- Ambigüidade produzida por padrões lingüísticos definidos de forma equivocada;
- Nível de conhecimento do engenheiro de requisitos em relação a classificação gramatical das palavras na Língua Portuguesa.

Com base nessas constatações, alteramos a proposta, de forma que continuaríamos utilizando os padrões lingüísticos, em um formato diferente, onde o engenheiro de requisitos não teria a liberdade de configurá-los, mas a ferramenta traria um conjunto pré-definido de padrões.

A definição desses padrões não é uma tarefa trivial, foram analisados vários textos em linguagem natural, para que fossem extraídos esses padrões lingüísticos. Algumas palavras atendem mais que uma regra, sendo necessária a definição de prioridades na aplicação dos padrões. A Tabela 3 apresenta a lista de padrões lingüísticos pré-definidos na sua ordem de prioridade.

A simples aplicação dos padrões listados na Tabela 3 não foi suficiente para identificarmos os elementos do modelo de classes, pois percebemos os seguintes problemas:

- Os textos de entrada em linguagem natural não poderiam conter acentuações, pois a utilização do modelo conceitual para uma futura implementação apresentaria problemas;
- Identificação da mesma classe, atributo e/ou operação com nomes distintos quando se tratava de palavras no plural e singular.
- Vínculo automático das classes com seus respectivos atributos e operações funcionou de maneira ineficiente.

Os problemas identificados no processo de utilização dos padrões lingüísticos foram solucionados da seguinte forma:

- Eliminação da acentuação em todas as palavras do texto em linguagem natural;

- No processo de classificação das palavras em classe, atributo e/ou operação, deve ser validado se essa palavra não está classificada para um dos tipos (classe, atributo, operação) no singular ou plural.
- Não vinculamos os atributos e operações de forma automática no modelo, permitindo que o engenheiro de requisitos realize essa tarefa através do processo manual de arrastar e soltar. Os atributos e operações são identificados e ficam à disposição do engenheiro de requisitos para o uso.

Tabela 3 – Definições de padrões lingüísticos da ferramenta PARADIGMA

Regra	Padrões Lingüísticos	Candidato à	Prioridade
R-01	**[**Conjunção OU Dois Pontos OU Vírgula**]** E **[**Substantivo E **[**Preposição+ Artigo OU Preposição**]** E Substantivo**]** E **[**Conjunção OU Vírgula OU Ponto OU Preposição + Artigo**]**	**Atributo**	**01**
R-02	**[**Conjunção OU Dois Pontos OU Vírgula**]** E **[**Nome Próprio OU Substantivo**]** E **[**Conjunção OU Vírgula OU Ponto OU Preposição + Artigo**]**	**Atributo**	**02**
R-03	Verbo E Artigo E **[**Substantivo OU Nome Próprio**]**	**Operação**	**03**
R-04	Substantivo E Adjetivo	**Classe com nome composto**	**04**
R-05	Substantivo	**Classe**	**05**
R-06	Palavras: um, uma (artigo indefinido).	**Multiplicidade 1**	**06**
R-07	Palavras: muitos, muitas, vários, várias, bastante, pelos, pelas.	**Multiplicidade ***	**07**
R-08	Classes que estão dentro da frase terminada por [Ponto] ou [Ponto e Vírgula]	**Associação**	**08**

A Tabela 4 apresenta as regras utilizadas para conversão do singular para o plural e vice-versa. Foram consideradas as regras de conversão para os substantivos. Com a aplicação das regras de conversão foi possível diminuir a identificação de elementos duplicados no modelo conceitual de classes.

Tabela 4 – Regras para conversão do singular para o plural - Substantivos

Regras	Ação
Regra geral	Acrescenta-se o “S”
Substantivos terminados em R, Z	Acrescenta-se “ES”
Substantivos terminados em AL, EL, OL, UL	Troca-se o “L” por “IS”
Substantivos terminados em IL	Os oxítonos: troca-se o “IL” por “IS”; paroxítonos: troca-se “IL” por “EIS”
Substantivos terminados em M	Troca-se o M por NS
Substantivos terminados em S	Monossílabos e oxítonos: acrescenta-se “ES”; não oxítonos ficam invariáveis
Substantivos terminados em X	Ficam invariáveis
Substantivos terminados em AO	Há três formas de plural: ÃOS, ÃES, ÕES

Na Tabela 5 são apresentados alguns exemplos da aplicação dos padrões lingüísticos dentro de um texto. Nos exemplos utilizam-se frases em linguagem natural. Logo em seguida, as frases são apresentadas com as marcações atribuídas pelo etiquetador de texto. Os fragmentos da frase onde são aplicadas as regras dos padrões lingüísticos são destacados e são descritas cada etiqueta das palavras ou símbolos que compõe estes fragmentos. De acordo com a estrutura sintática dos fragmentos, são aplicadas as regras em ordem de prioridade. Após a aplicação das regras obtêm-se os elementos do modelo de classes conceitual, tais como: classes, atributos, operações, associações, multiplicidades.

Tabela 5 – Exemplos - Aplicação de padrões lingüísticos no texto

Exemplos - Padrões lingüísticos			
Frase ➔	Na ficha de cadastro deve constar: nome do cliente, endereço, data de cadastro, telefone e email.		
Frase com Marcações ➔	Na_PREP+ART ficha_N de_PREP cadastro_N deve_VERB constar_VERB :_: nome_N do_PREP+ART cliente_N ,_, endereço_N ,_, data_N de_PREP cadastro_N ,_, telefone_N e_CONJ email_N ._.		
Fragmento da frase	**Marcadores em cada palavra / símbolo**	**Regra Aplicada**	**Resultado**
Ficha	ficha_N[Substantivo]	R-05[Classe]	FICHA
Cadastro	cadastro_N[Substantivo]	R-05[Classe]	CADASTRO
: nome do cliente,	:_:[Dois Pontos] nome_N[Substantivo] do_PREP+ART[Preposição+Artigo] cliente_N [Substantivo] ,_,[Vírgula]	R-01[Atributo]	NOME_DO_CLIENTE
, endereço,	,_,[Vírgula] Endereço_N[Substantivo] ,_,[Vírgula]	R-02[Atributo]	ENDERECO
, data de cadastro,	,_,[Vírgula] data_N[Substantivo] de_PREP[Preposição] cadastro_N[Substantivo] ,_,[Vírgula]	R-01[Atributo]	DATA_DE_CADASTRO
, telefone e	,_,[Vírgula] telefone_N[Substantivo] e_CONJ[Conjunção]	R-02[Atributo]	TELEFONE
e email.	e_CONJ[Conjunção] email_N[Substantivo] ._.[Ponto]	R-02[Atributo]	EMAIL
Frase ➔	Uma loja de CDs possui CDs para venda e para locação.		
Frase com Marcações ➔	Uma_ART loja_N de_PREP CDs_NP possui_VERB CDs_NP para_PREP venda_N e_CONJ para_PREP locação_N ._.		
Fragmento da frase	**Marcadores em cada palavra / símbolo**	**Regra Aplicada**	**Resultado**
Loja	loja_N[Substantivo]	R-05[Classe]	LOJA
Venda	venda_N[Substantivo]	R-05[Classe]	VENDA
Locação	locação_N[Substantivo]	R-05[Classe]	LOCACAO
Uma loja ...para venda	Associação/Multiplicidade Uma_ART[Artigo] Loja_N[Substantivo] ... Venda_N[Substantivo]	R-08 [Associação] e R-06 [Multiplicidade 1]	1 Loja – Associação -- Venda
Frase ➔	receber e conferir o pagamento efetuado pelos clientes;		
Frase com Marcações ➔	receber_VERB e_CONJ conferir_VERB o_ART pagamento_N efetuado_ADJ pelos_PREP+ART clientes_N ;_;		
Fragmento da frase	**Marcadores em cada palavra / símbolo**	**Regra Aplicada**	**Resultado**
conferir o pagamento	conferir_VERB[Verbo] o_ART[Artigo] pagamento_N[Substantivo]	R-03 [Operação]	CONFERIR_PAGAMENTO
pagamento efetuado	pagamento_N[Substantivo] efetuado_ADJ[Adjetivo]	R-04 [Classe com nome composto]	PAGAMENTO_EFETUADO
Clientes	clientes_N[Substantivo]	R-05 [Classe]	CLIENTES

4.2 MODELAGEM DA FERRAMENTA

A documentação da ferramenta PARADIGMA foi desenvolvida utilizando a UML e os conceitos da orientação a objetos. Os principais modelos diagramáticos que representam a ferramenta são apresentados nas subseções a seguir.

4.2.1 DIAGRAMA DE CASOS DE USO

O diagrama de casos de uso fornece um modo de descrever a visão externa do sistema e suas interações com o mundo exterior, representando uma visão de alto nível da funcionalidade do sistema, mediante a interação dos atores. Na Figura 9 é apresentado o diagrama de casos de uso da ferramenta PARADIGMA. O Engenheiro de requisitos é o principal ator, ele realiza a interação com a ferramenta para obter como resultado final o modelo conceitual de classes.

No caso de uso UC-001, o engenheiro de requisitos seleciona o arquivo texto, no formato ASCII, com as descrições dos requisitos em linguagem natural. Esse arquivo não poderá conter informações diferentes de texto, tais como: gráficos, imagens, tabelas, etc.

No caso de uso UC-002, o arquivo texto é ajustado para o formato que o etiquetador de texto necessita e execute o caso de uso UC-003, onde a etiquetagem do texto é realizada, gerando um arquivo com as marcações (*tags*).

No caso de uso UC-004, o arquivo com as marcações é processado e as regras dos padrões lingüísticos são aplicadas às palavras do texto de acordo com sua classificação gramatical. O caso de uso UC-005 realiza a identificação das palavras candidatas à classe de acordo com as regras estipuladas no padrões lingüísticos. O caso de uso UC-006 realiza a identificação das palavras candidatas a atributo de acordo com as regras estipuladas no padrões lingüísticos. O caso de uso UC-007 realiza a identificação das palavras candidatas à operação de acordo com as regras estipuladas no padrões lingüísticos. Esses três casos de uso executam o caso de uso UC-008, onde são realizadas as conversões das palavras do singular para o plural e vice-versa,

para evitar a identificação da mesma classe, atributo ou operação no singular e plural.

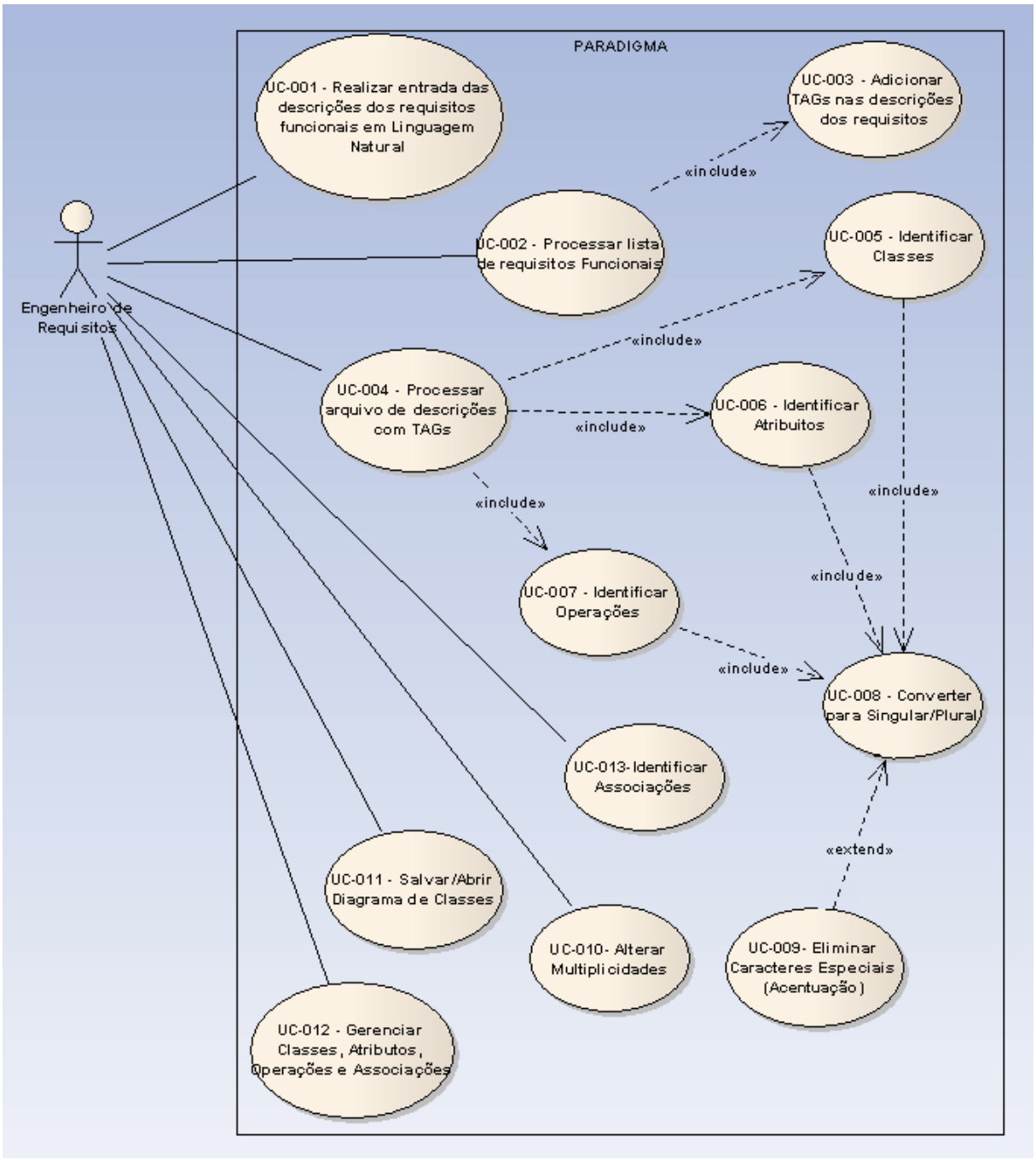


Figura 9 – Diagrama de Casos de Uso da ferramenta PARADIGMA

Quando necessário, o UC-009 é executado. Esse caso de uso permite a eliminação dos caracteres especiais do textos, mais especificamente, a acentuação e o caracter especial “ç”.

No caso de uso UC-013, é realizada a identificação das associações. Essa identificação leva em consideração a relação existente em

candidatas a classe dentro de uma frase, nesse mesmo caso de uso, as multiplicidades são definidas. O engenheiro de requisitos executa o caso de uso UC-010, quando necessita alterar as multiplicidades identificadas automaticamente pela ferramenta.

O caso de uso UC-011 permite ao engenheiro de requisitos abrir e/ou salvar os modelos conceituais de classes gerados pela ferramenta. No caso de uso UC-012 é onde o engenheiro de requisitos pode interagir com a interface gráfica da ferramenta, manipulando todos os elementos do modelos de classes. Nesse caso de uso ele poderá adicionar, excluir, alterar classes, atributos, operações e associações.

4.2.2 DIAGRAMA DE CLASSES

A ferramenta PARADIGMA foi desenvolvida no contexto da Orientação a Objetos e foi utilizado o diagrama de classes da UML para descrever as classes e os relacionamentos entre elas. A Figura 10 apresenta a visão conceitual das classes e a Figura 11 apresenta a visão de implementação das classes da ferramenta.

As classes de *interfaces* da ferramenta são herdadas da classe Tform, a classe que serve de base para o desenho do diagrama é herdada do Tpanel e as classes de desenho são herdadas do Tpaintbox. As classes identificadas no diagrama de classes da ferramenta PARADIGMA são descritas a seguir:

- *TfrmMain*: tela principal da ferramenta, onde estão os *menus* e barra de ferramentas para criação dos elementos do diagrama de classes. Através dessa classe que é iniciado todo o processo de captura e processamento dos requisitos em linguagem natural. São executados os métodos de aplicação das regras dos padrões lingüísticos, conversão de singular/plural e eliminação dos caracteres especiais (acentuação).
- *TfrmTextoRequisito*: formulário no qual são editadas as descrições das classes, atributos e operações. Permite a adição de novos atributos e operações a uma determinada classe.
- *TfrmAbrirModelo*: faz o mapeamento das tabelas da base de dados para as classes da ferramenta e instancia os objetos.

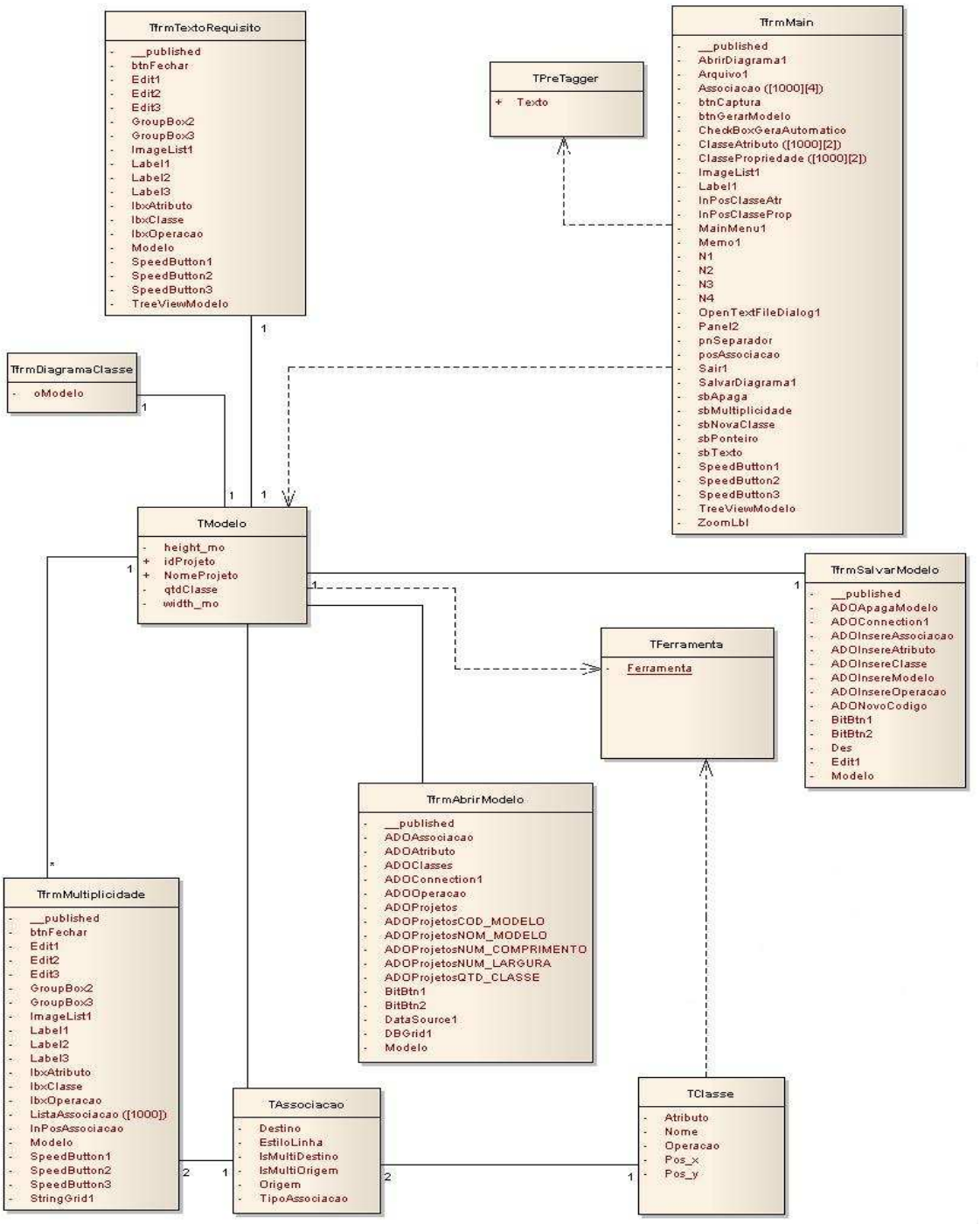


Figura 10 – Diagrama de Classes Conceitual

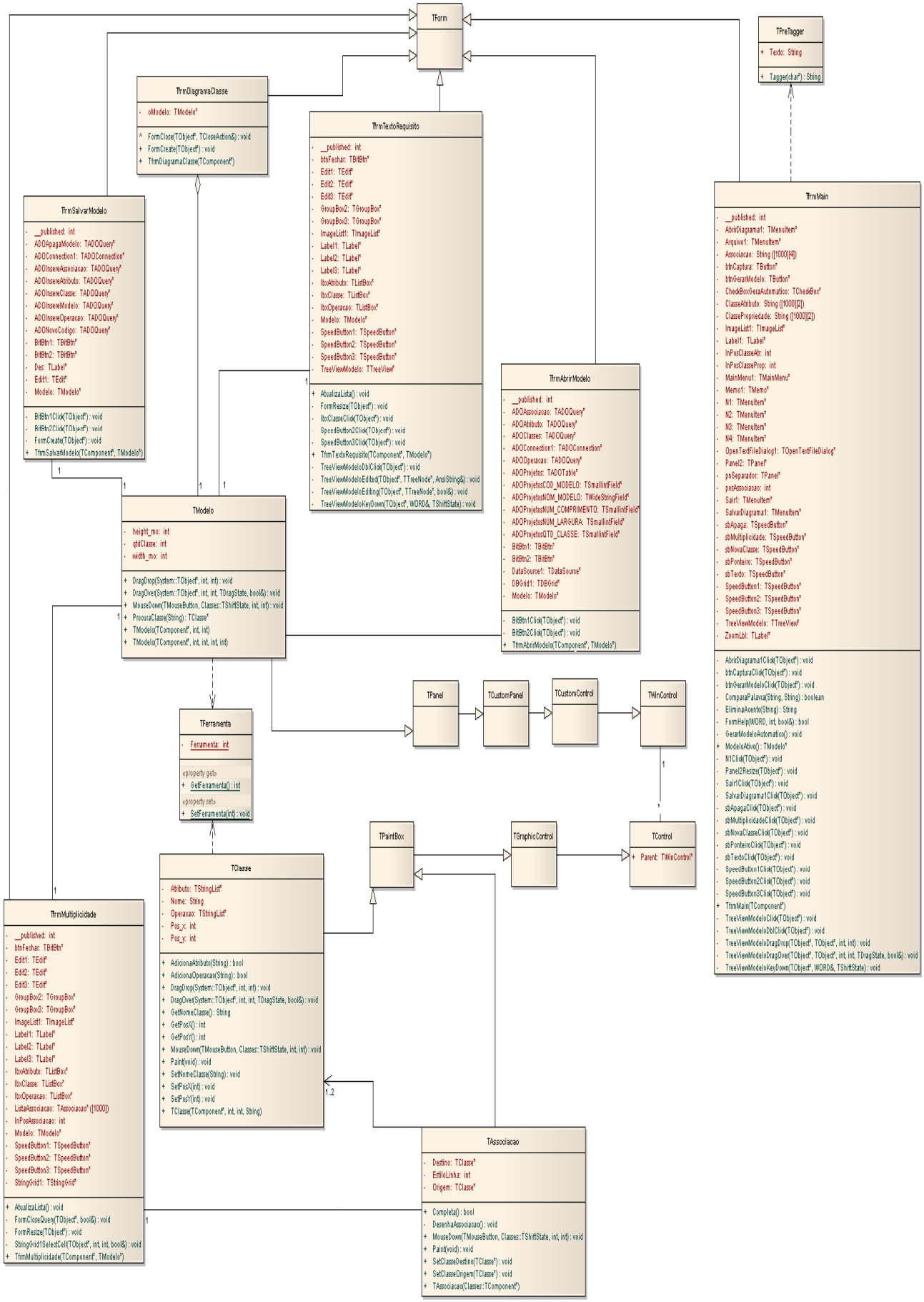


Figura 11 – Diagrama de Classes de Implementação

- *TfrmSalvarModelo*: faz o mapeamento das classes para as tabelas na base de dados e persiste os dados dos objetos em memória nas tabelas.
- *TfrmMultiplicidade*: formulário no qual o engenheiro de requisitos altera as multiplicidades das associações existentes entre as classes.
- *TfrmDiagramaClasse*: classe onde é apresentado o diagrama de classes na forma visual. Essa classe poderá ter várias instâncias, possibilitando ao engenheiro de requisitos, a manipulação de vários diagramas simultaneamente.
- *TPreTagger:* classe responsável por formatar o arquivo de entrada no formato exigido pelo etiquetador de texto.
- *TFerramenta:* classe que controla qual ferramenta está selecionada para manipulação do ambiente gráfico (selecionar, adicionar classe, excluir classe, adicionar associação, etc.)
- *TModelo:* classe que serve de base para toda a parte visual da ferramenta PARADIGMA. Essa classe dá suporte, em termos visuais, para a criação dos demais objetos gráficos (classe e associações).
- *TClasse:* classe que permite a apresentação de uma classe do diagrama no formato visual. É possível, através dessa classe, realizar a adição dos atributos e operações no diagrama visualmente. Utilizar os recursos de arrastar e soltar no objeto visual na tela de edição.
- *TAssociação:* classe que permite a apresentação visual de um relacionamento entre classes no diagrama.

4.2.3 DIAGRAMA ENTIDADE RELACIONAMENTO

Algumas classes utilizadas na ferramenta PARADIGMA deverão ser persistidas e a proposta foi utilizar um banco de dados relacional para persistir essas informações. Foi necessário realizar o mapeamento entre as classes e as tabelas da base de dados onde serão armazenadas. O processo inverso no momento de abrir um diagrama de classes salvo em tabela também foi necessário.

A seguir, na Figura 12, apresenta-se o DER (Diagrama Entidade Relacionamento) da ferramenta PARADIGMA.

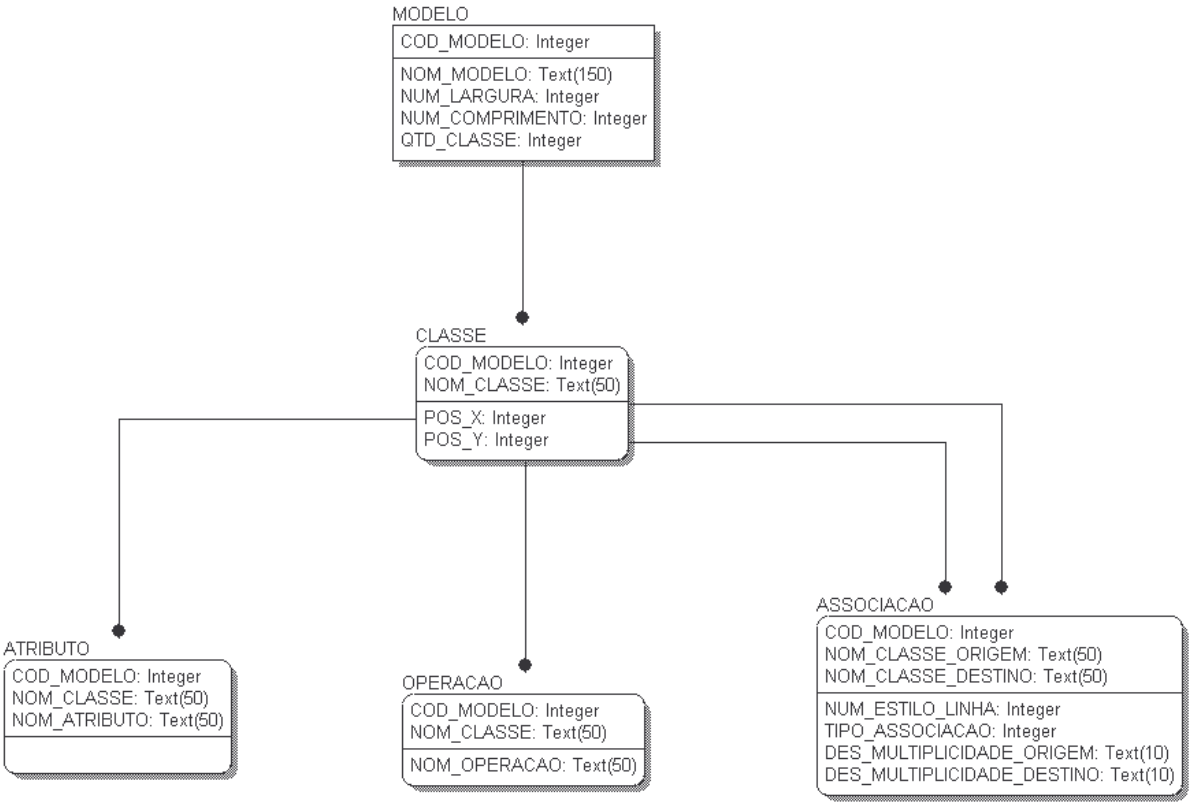


Figura 12 – Diagrama de entidade e relacionamento da ferramenta PARADIGMA

4.3 INTERFACES E FUNCIONALIDADES DA FERRAMENTA

O Apêndice B traz as *interfaces* da ferramenta PARADIGMA e algumas notas explicativas referentes às mesmas. As funcionalidades implementadas na ferramenta e que estão disponíveis para o engenheiro de requisitos utilizar são:

- Adicionar marcações no texto de entrada em linguagem natural;
- Analisar e aplicar os padrões lingüísticos definidos previamente e classificar as palavras nos seguintes tipos: classes, atributos, operações e associações;
- Exibir o texto de entrada, no momento de captura, para facilitar o rastreamento pelo engenheiro de requisitos, para confirmar se a ferramenta classificou as palavras nos tipos corretos;
- Alterar e excluir palavras que estão fora do domínio da aplicação;

- Vincular os atributos e operações às suas classes respectivas;
- Gerar modelo conceitual de classes automaticamente com base nos dados capturados pela ferramenta;
- Alterar e/ou adicionar classes, atributos e operações no modelo de classes gerado;
- Adicionar associações entre as classes;
- Excluir classes, atributos, operações e associações do modelo de classes;
- Adicionar o relacionamento de generalização, caso necessário. Vale ressaltar, que a ferramenta não identifica automaticamente esse tipo de relacionamento, mas dá suporte a ele.
- Alterar as multiplicidades das associações;
- Salvar/Abrir o modelo de classe conceitual gerado pela ferramenta;
- Organizar o modelo de classes graficamente, através do recurso de arrastar e soltar.

4.4. COMPARATIVO ENTRE FERRAMENTAS QUE UTILIZAM PLN

O estudo comparativo entre as ferramentas NL-OOPS, DIPETT-HAIKU, CM-BUILDER, LIDA RCR e GOOAL apresentado nas Tabelas 6, 7 e 8 foi realizado por Li, Dewar e Pooley (2005).

Li, Dewar e Pooley (2005) salientam que o código-fonte ou versões funcionais das ferramentas não foram analisados diretamente, por razões variadas e que o quadro comparativo obtido foi produzido com base nos artigos escritos e publicados pelos autores.

A partir do estudo comparativo realizado por Li, Dewar e Pooley (2005), adicionamos a ferramenta PARADIGMA, desenvolvida neste trabalho. Esse estudo abrange ferramentas desenvolvidas em projetos que propõem a

utilização do PLN, onde recebem como entrada textos em linguagem natural e geram diagramas que dão suporte ao paradigma OO.

Na Tabela 6 são apresentados os tipos de diagramas que cada ferramenta se propõe a gerar como saída. Os diagramas que a maioria das ferramentas geram são os diagramas de classes e objetos e a ferramenta PARADIGMA se encaixa nesse contexto. O diagrama de casos de uso é provido somente pela ferramenta RECORD, e GOOAL é a única que provê o diagrama de sequência. Fica evidente a necessidade de pesquisas para atender a visão dinâmica dos sistemas em desenvolvimento, a maioria se especializou em atender a visão estática.

Tabela 6 – Modelagem OO automática utilizando PLN

Tipo de Diagrama	PARADIGMA	NL-OOPS	DIPETT-HAIKU	CM-BUILDER	LIDA	RECORD	GOOAL
Diagrama Casos de Uso	Não	Não	Não	Não	Não	Sim	Não
Diagrama de Classes	Sim	Não	Não	Sim	Sim	Não	Sim
Diagrama de Objetos	Não	Sim	Sim	Não	Sim	Sim	Não
Diagrama de Seqüência	Não	Não	Não	Não	Não	Não	Sim
Diagrama de Atividade	Não	Não	Não	Não	Não	Não	Não

Na Tabela 7 são analisadas as ferramentas que geram o diagrama de classes como saída e os repectivos métodos utilizados para chegar a esse diagrama.

A ferramenta LIDA exige uma grande interação do engenheiro de requisitos em todo o processo de definição do diagrama de classes. As demais ferramentas realizam boa parte do processo de forma automatizada, permitindo a interação do engenheiro de requisitos após a geração do diagrama de classes inicial.

A ferramenta PARADIGMA permite a geração automatizada do diagrama, aplicando as regras identificadas na estrutura de escrita dos requisitos em linguagem natural para a Língua Portuguesa, os quais chamamos de padrões lingüísticos. Foram identificados oito regras para identificação de classes, atributos, operações, relacionamentos e multiplicidades (Seção 4.1.3). Após essa identificação inicial, o engenheiro de requisitos poderá interagir com o diagrama realizando os ajustes necessários, utilizando a edição gráfica da própria ferramenta PARADIGMA.

Tabela 7 – Identificando classes utilizando PLN

	PARADIGMA	CM-BUILDER	LIDA	GOOAL
Candidato à Classe	R-04/R-05 [Padrões Lingüísticos]	Substantivo	Engenheiro de Requisitos	Substantivo
Identificando Classes	Palavras que atendem a R-04/R-05 na ordem de prioridade	Freqüência do substantivo no texto	Engenheiro de Requisitos	Técnica de análise lingüística
Identificando Associações	R-08 [Padrões Lingüísticos]	Sujeito/ Verbo/ Objeto	Engenheiro de Requisitos	Linguagem Natural semi-estruturada
Multiplicidade	R-06/R-07 [Padrões Lingüísticos]	Utiliza os determinantes indefinidos, artigos indefinidos ou artigos definidos com substantivo no singular	Engenheiro de Requisitos	Não
Identificando Atributos	R-01/R-02 [Padrões Lingüísticos]	Sim	Engenheiro de Requisitos	Sim
Identificando Operações	R-03 [Padrões Lingüísticos]	Não	Engenheiro de Requisitos	Sim

A ferramenta CM-BUILDER basea-se na freqüência com que cada substantivo aparece no texto para selecionar as palavras candidatas às classes. HARMAIN e GAIZAUSKAS (2000) resumem em 10 passos, o processo necessário, para partir dos textos já etiquetados até o diagrama de classes (geração automatizada). Esse diagrama inicial é gerado em um formato que permite a

importação para uma ferramenta CASE para que o engenheiro de requisitos possa interagir no diagrama gerado.

A ferramenta GOOAL utiliza uma linguagem natural semi-estruturada. As descrições dos requisitos em linguagem são traduzidas para a linguagem semi-estruturada 4W, a partir dessa linguagem, a ferramenta identifica os elementos do diagrama. O engenheiro de requisitos pode interagir nos modelos gerados para realizar as correções necessárias.

As ferramentas capazes de identificar os relacionamentos entre as classes de forma automática são: PARADIGMA, CM-BUILDER e a GOOAL. Na Tabela 8, são apresentados os tipos de relacionamentos que cada ferramenta é capaz de identificar automaticamente.

A associação e suas multiplicidades são identificadas por todas as ferramentas. A agregação é identificada somente pela ferramenta CM-BUILDER. A composição, generalização e dependência não são identificadas por nenhuma das ferramentas.

Os relacionamentos estão diretamente ligados à estrutura de escrita da linguagem natural. Para a correta identificação dos relacionamentos de agregação, composição, generalização e dependência, deveriam ser seguidos alguns padrões estruturados de escrita.

Alguns padrões de escrita que permitiria a identificação de alguns relacionamentos:

- **Agregação:** “... pertence a...”
- **Generalização: “**... é um tipo de...**”**
- **Composição:** “... é parte de...”
- **Dependência:** “... utiliza...” ou “... depende de...”

Como os textos de entrada em linguagem natural não são descritos seguindo essas regras de padrões estruturados, a identificação desses tipos de relacionamentos acaba sendo realizada pela intervenção do engenheiro de requisitos. A ferramenta PARADIGMA permite representar no modelo conceitual de classes, na forma gráfica, somente os seguintes tipos de relacionamentos: associação e generalização.

Tabela 8 – Identificando relacionamentos entre classes utilizando PLN

	PARADIGMA	CM-BUILDER	GOOAL
Associação	Sim	Sim	Sim
Multiplicidade na Associação	Simples/Múltipla	Simples/Múltipla	Simples/específica
Agregação	Não	Sim	Não
Composição	Não	Não	Não
Generalização	Não	Não	Não
Dependência	Não	Não	Não

Diferencial da ferramenta PARADIGMA em relação às ferramentas similares:

- **Suporte à Língua Portuguesa:** As ferramentas GOOAL e CM-BUILDER dão suporte à Língua Inglesa.
- **Utiliza linguagem natural pura:** A ferramenta GOOAL necessita realizar a tradução da linguagem natural para 4W, uma linguagem semi-estruturada.
- **Permite a interação gráfica pelo engenheiro de requisitos diretamente no processo de geração do diagrama:** A ferramenta CM-BUILDER necessita uma ferramenta CASE de terceiros.
- **Definição dos padrões lingüísticos:** A ferramenta GOOAL substitui essa opção pela linguagem semi-estruturada e a CM-BUILDER aplica algumas regras, similares ao que chamamos de padrões lingüísticos.

5 EXPERIMENTAÇÃO DA FERRAMENTA PARADIGMA

A experimentação da ferramenta PARADIGMA demonstrou-se uma atividade de suma importância. Através dessa atividade, procuramos identificar se a ferramenta atingiu o objetivo proposto de auxiliar o engenheiro de requisitos, através da geração do modelo conceitual de classes, a partir dos requisitos em linguagem natural. Foi possível mensurar, de forma preliminar, qual a efetiva contribuição da ferramenta no contexto da Engenharia de Requisitos.

Foram avaliadas algumas opções para a realização dessa experimentação e, ao final, escolhemos a opção descrita a seguir:

- ✓ Utilizar engenheiros de requisitos com experiência para realizar o experimento, fornecendo um roteiro de utilização da ferramenta, para que dentro desse ambiente controlado pudéssemos avaliar os resultados obtidos. Esses engenheiros de requisitos poderiam ser professores da área de Engenharia de Software, Engenharia de Requisitos ou profissionais de mercado que atuam no segmento de Engenharia de Requisitos.

5.1 PARTICIPANTES

Realizamos uma pré-seleção das pessoas que se enquadravam no perfil desejado para a realização da experimentação da ferramenta PARADIGMA. Desses possíveis participantes, 5 eram professores com experiência nas disciplinas de Engenharia de *Software* e/ou Engenharia de Requisitos e 6 eram profissionais que atuam como engenheiros de requisitos, com pelo menos 3 anos de experiência na área, totalizando 11 pré-selecionados.

Foi enviado um *e-mail* para esse grupo de pessoas pré-selecionadas, convidando-as a participar de uma pesquisa para a avaliação da ferramenta PARADIGMA. Nesse *e-mail,* havia uma explicação rápida da finalidade da ferramenta e, em anexo, foi enviado o cenário de utilização da ferramenta (vide Apêndice A).

Os participantes deveriam seguir o cenário de utilização, onde constava a descrição funcional do *software*, as atividades a serem realizadas, o questionário para avaliar a usabilidade da ferramenta e um espaço para sugestões de melhorias na ferramenta.

Desse grupo de 11 pessoas convidadas, somente 36% participaram de todo o processo proposto para a experimentação da ferramenta, totalizando 4 participantes, sendo 2 da área acadêmica e 2 profissionais de mercado. A seguir, é apresentado o cenário de utilização aplicado na experimentação e os resultados obtidos.

5.2 Cenário de Utilização da Ferramenta PARADIGMA

Os participantes receberam um roteiro para realização do cenário contendo todos os passos necessários para o processo de experimentação.

A descrição funcional em linguagem natural, utilizada no cenário de utilização, é referente a um software para uma loja de CDs (vide Apêndice A). Utilizando essa descrição funcional, os participantes deveriam elaborar o modelo conceitual de classes da forma convencional e desenhar esse modelo em uma ferramenta CASE de sua preferência (as ferramentas utilizadas foram: JUDE, Enterprise Architect).

Todo o procedimento de modelagem do modelo conceitual de classes, de maneira manual, deveria ser cronometrado. A preferência é que essa atividade não sofresse interrupção, caso houvesse interrupção, os participantes deveriam anotar todos os tempos envolvidos na modelagem para que pudéssemos chegar a um tempo total final.

A partir do modelo conceitual de classes desenvolvido pelo participante, o mesmo deveria realizar o próximo passo do cenário de utilização. O passo seguinte foi gerar o modelo conceitual de classes com o auxílio da ferramenta PARADIGMA. Caso o modelo gerado de forma automática pela ferramenta apresentasse alguma divergência do modelo desenvolvido pelo participante, ele deveria realizar os ajustes necessários para chegar ao modelo “correto”. Essa atividade deveria ser cronometrada também.

Depois da realização das atividades de modelagem do modelo conceitual de classes manualmente e com a ajuda da ferramenta (automatizada), os participantes deveriam responder um questionário referente à utilização da

ferramenta e caso desejassem, contribuir indicando melhorias necessárias na ferramenta PARADIGMA através do item sugestões.

Utilizamos os resultados obtidos através da aplicação do cenário de utilização para promover as seguintes avaliações:

- Comparar o modelo conceitual de classes gerado pelo engenheiro de requisitos com o modelo conceitual de classes gerado automaticamente pela ferramenta PARADIGMA;
- Analisar o tempo gasto e as dificuldades encontradas para chegar ao modelo "correto", através do processo tradicional e com o auxílio da ferramenta PARADIGMA;
- Comparar os modelos conceituais de classes gerados pelos participantes sem a utilização da ferramenta;
- Avaliar a relevância da ferramenta no contexto da Engenharia de Requisitos.

5.3 Resultados da Experimentação

Todos os participantes utilizaram a mesma descrição funcional de software para realizar a atividade de modelagem do modelo conceitual de classes. Esse modelo de classes foi criado de forma convencional, sem o auxílio da ferramenta PARADIGMA, e posteriormente, foi modelado com o auxílio da ferramenta.

A seguir, são apresentados os modelos que cada participante considerou como correto e o modelo gerado automaticamente pela ferramenta. Iremos apresentar um modelo de classes padrão, que será a fusão dos modelos dos participantes, levando em consideração a interseção dos modelos, ou seja, os elementos comuns a todos. Os participantes são identificados durante a exposição e análise dos resultados como Participante 1 (P1), Participante 2 (P2), Participante 3 (P3) e Participante 4 (P4).

5.3.1 Modelo Conceitual de Classes Gerado pela Ferramenta PARADIGMA

O modelo conceitual de classes gerado automaticamente pela ferramenta PARADIGMA, com base na descrição funcional do *software* para a loja

de CDs é mostrado na Figura 13. Esse modelo não sofreu intervenção humana, sendo mostrado exatamente como a ferramenta o gerou.

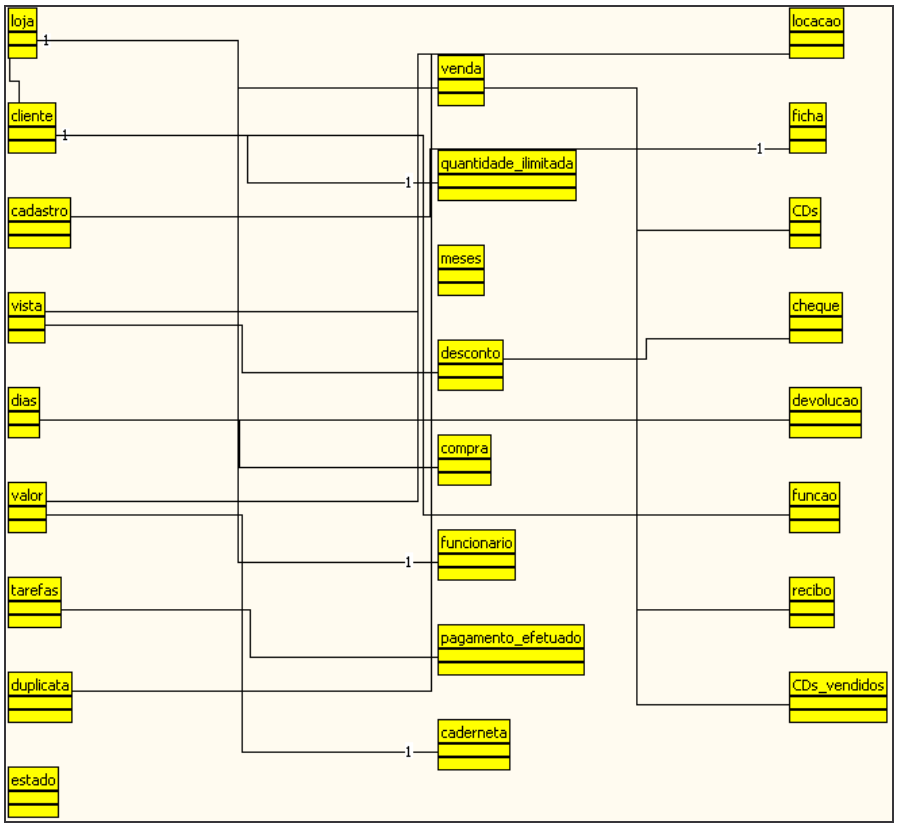


Figura 13 – Modelo conceitual de classes gerado pela ferramenta

Na Figura 14, o modelo de classes gerado pela ferramenta foi organizado para facilitar a visualização das classes e respectivos relacionamentos. Do lado direito da figura, são apresentados os atributos e operações identificados automaticamente pela ferramenta. Esses atributos e operações foram identificados, mas não foram vinculados às respectivas classes, essa atividade deve ser realizada pelo engenheiro de requisitos. O modelo que a ferramenta gera, a partir de uma determinada descrição funcional ou texto em linguagem natural sempre é o mesmo, não havendo variação entre a experimentação realizada pelos diferentes participantes.

A ferramenta identificou automaticamente 25 classes, 18 associações, 5 atributos e 6 operações baseando-se nos padrões lingüísticos pré-definidos. Em 5 das 18 associações foram identificadas multiplicidades.

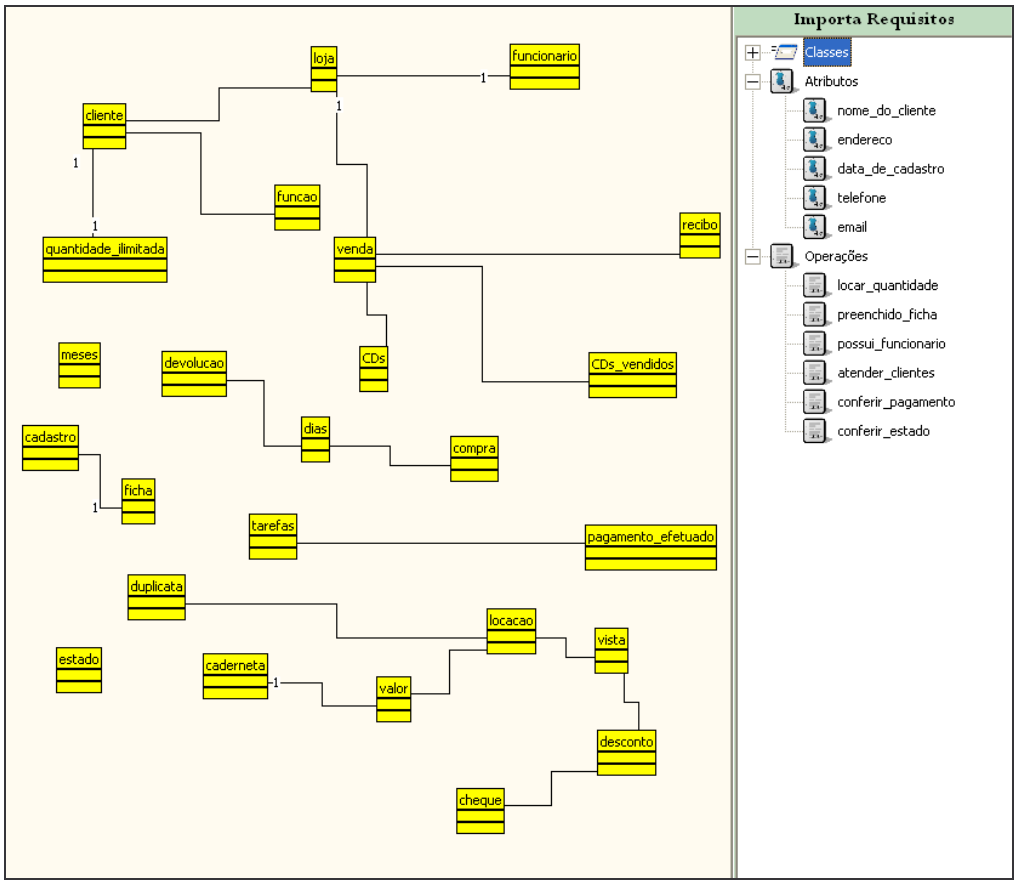


Figura 14 – Modelo conceitual de classes gerado pela ferramenta, organizado com intervenção humana.

5.3.2 Modelos Conceituais de Classes Gerados pelos Participantes

Os modelos conceituais de classes que os participantes modelaram são apresentados nessa subseção. Ainda não nos preocuparemos em realizar as comparações entre os modelos feitos pelo modelador humano e o modelo de classes gerado automaticamente pela ferramenta PARADIGMA.

Na Figura 15, é apresentado o modelo conceitual de classes elaborado pelo Participante 1. Percebemos que este participante utilizou o relacionamento de generalização, que não é contemplado no processo de geração automática do modelo de classes. Outro detalhe que chamou a atenção foi o fato de não haver a identificação de atributos e operações para nenhuma das classes do modelo. As multiplicidades entre as classes também não foram todas identificadas e devidamente adicionadas ao modelo. Como se trata de um modelo conceitual está dentro da expectativa.

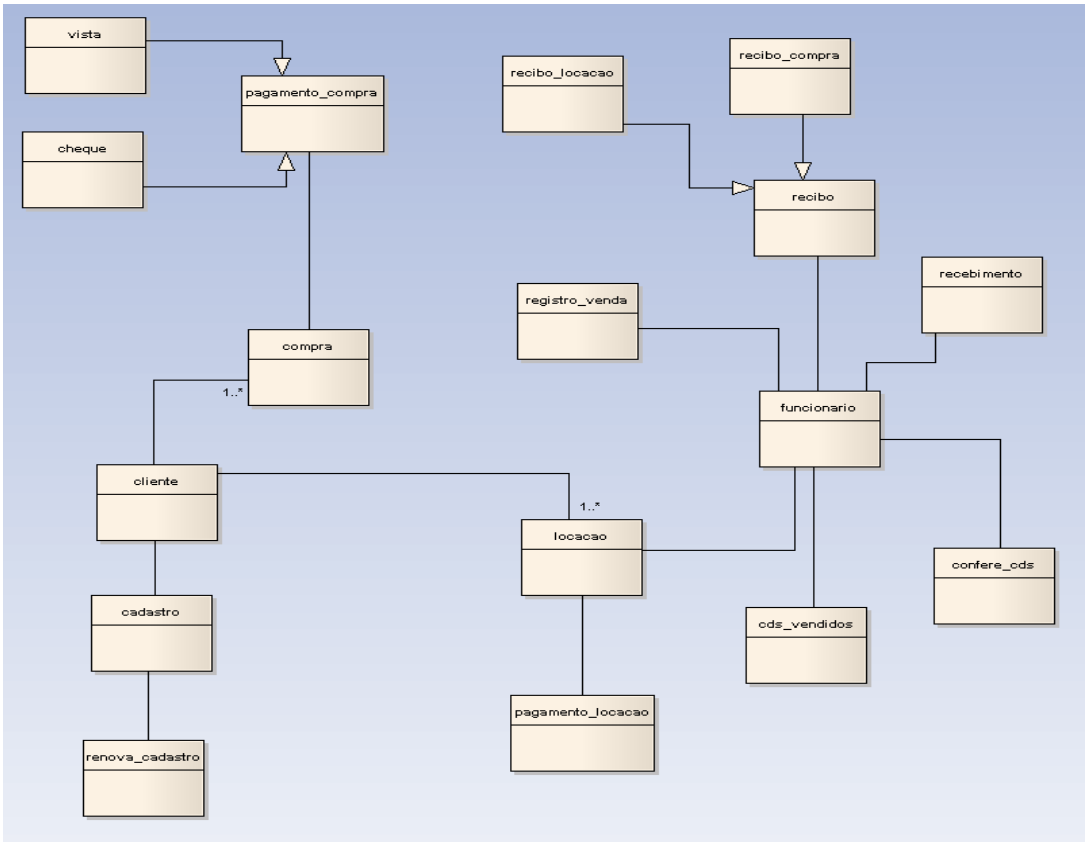


Figura 15 – Modelo Conceitual de Classes – Participante 1

Na Figura 16, é apresentado o modelo conceitual de classes do Participante 2. Nesse modelo podemos perceber que foram identificados alguns atributos e há um número significativamente menor de classes, ou elementos.

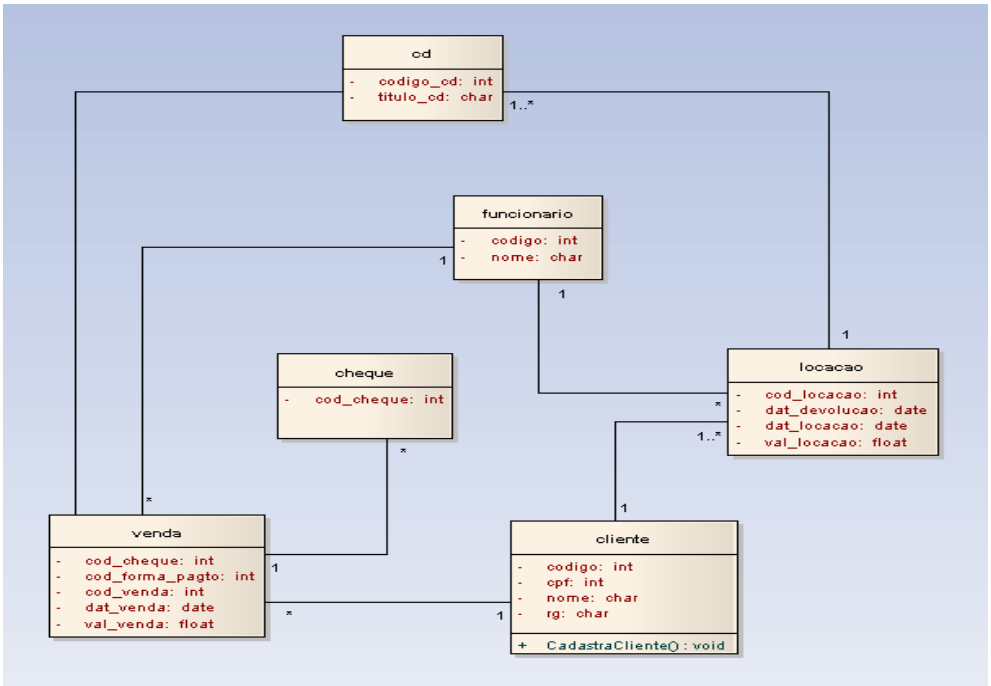


Figura 16 – Modelo Conceitual de Classes – Participante 2

O Participante 3 gerou o modelo conceitual de classes apresentado na Figura 17. Nesse modelo, podemos perceber que o participante acabou identificando um conjunto de atributos que não constatava na descrição funcional do software proposto. Geralmente acontece esse fenômeno, quando o engenheiro de requisitos já conhece o domínio da aplicação e consegue definir alguns elementos do modelo a partir dos seus conhecimentos prévios.

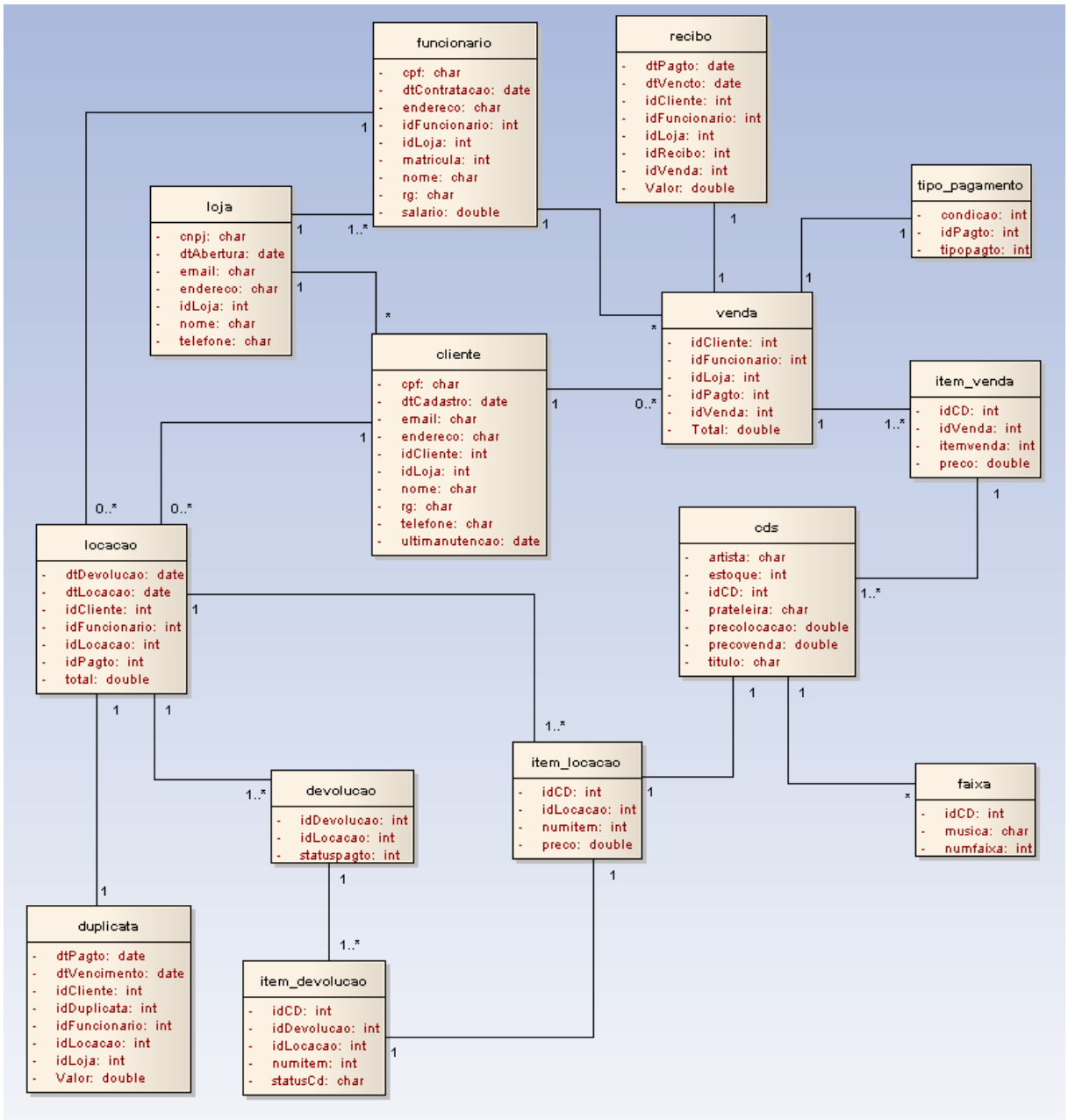


Figura 17 – Modelo Conceitual de Classes – Participante 3

Temos na Figura 18, a representação do modelo conceitual de classes gerado pelo Participante 4. Os modelos apresentados são utilizados para

realizar a comparação visual com o diagrama de classes gerado automaticamente pela ferramenta PARADIGMA.

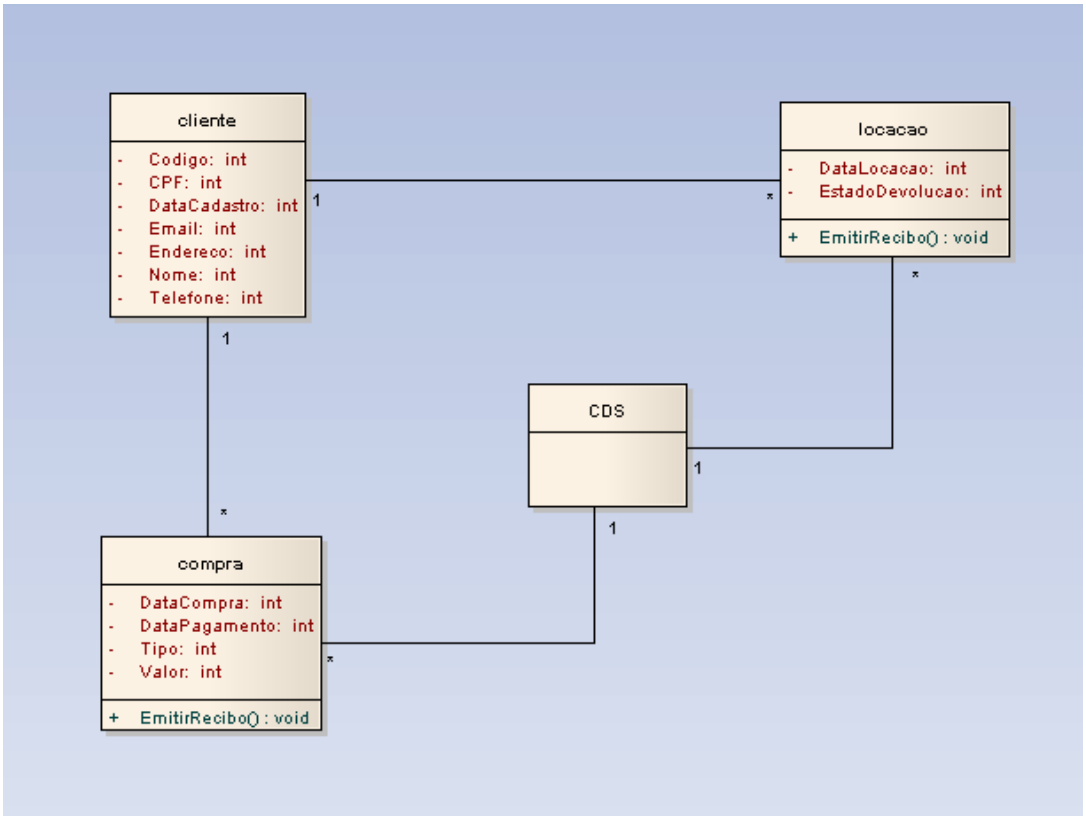


Figura 18 – Modelo Conceitual de Classes – Participante 4

5.3.3 Comparativo entre os Modelos dos Participantes e da Ferramenta

Os modelos de classes gerados pelos participantes apresentam diferenças significativas entre si, o que nos levou a realizar um comparativo para cada participante. Comparamos os modelos gerados pelos participantes, com o modelo gerado automaticamente pela ferramenta PARADIGMA. Nesse comparativo, consideramos os seguintes elementos do modelo: classes, atributos, operações e relacionamentos. Foram totalizados os elementos identificados somente pela ferramenta, somente pelo participante e os identificados tanto pelo participante quanto pela ferramenta.

Na Figura 19, visualizamos o comparativo entre o modelo gerado pelo P1 com o modelo gerado pela ferramenta PARADIGMA. Foram identificadas 33 classes, onde 16 delas, foram identificadas somente pela ferramenta PARADIGMA, 8 foram identificadas somente pelo P1 e 9 classes foram identificadas por ambos, ou seja, 53% das classes do modelo gerado pelo P1 foram contempladas no modelo

gerado pela ferramenta. Os atributos e operações não foram considerados pelo P1, não existindo base para a comparação entre os modelos, somente a ferramenta identificou atributos e operações. Quanto aos relacionamentos, 100% deles foram divergentes, ou seja, nenhum dos relacionamentos feitos pelo P1 coincidiu com os apresentados pela ferramenta PARADIGMA.

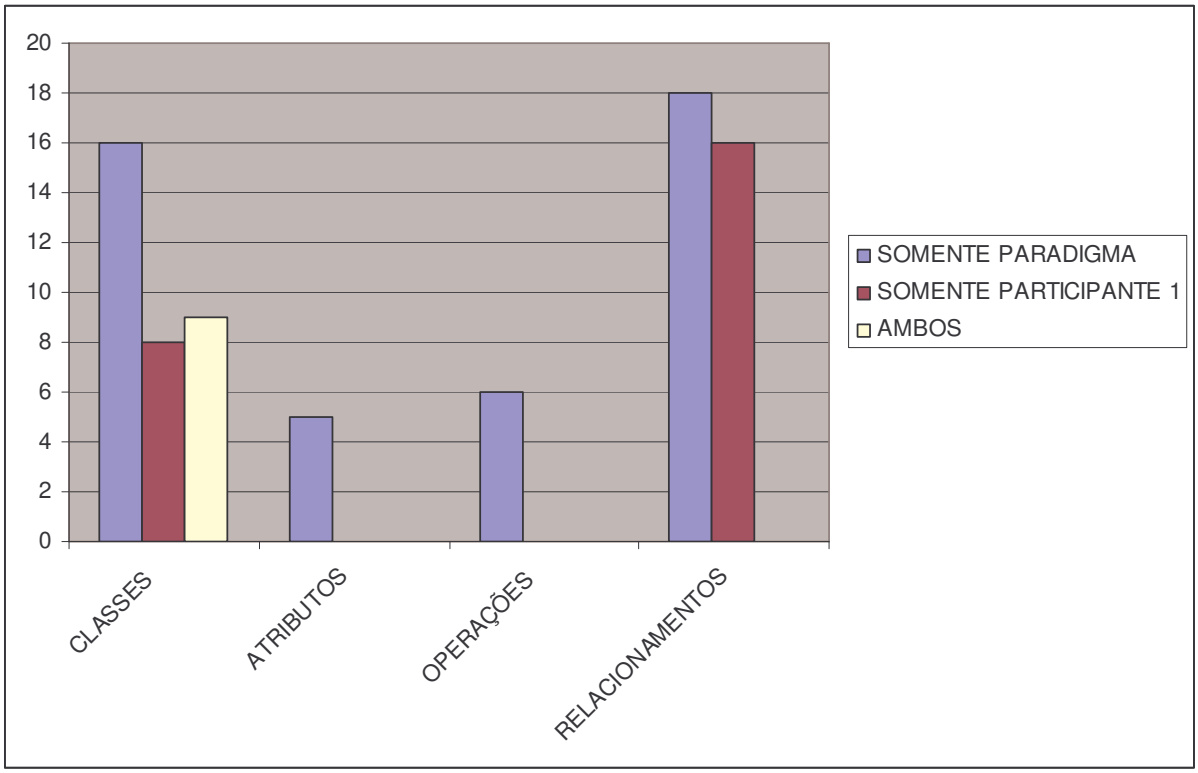


Figura 19 – Gráfico comparativo entre o modelo do P1 e o da ferramenta

Na Figura 20, é apresentado o comparativo entre o modelo gerado pelo P2 com o modelo gerado pela ferramenta PARADIGMA. Foram identificadas 25 classes, onde 19 classes foram identificadas somente pela ferramenta PARADIGMA e 6 classes foram identificadas por ambos. Nesse caso, 100% das classes do modelo gerado pelo P2 foram contempladas pelo modelo gerado automaticamente. Foram identificados 20 atributos, nos quais 4 atributos foram identificados somente pela ferramenta; 15 atributos foram identificados somente pelo P2 e 1 atributo foi identificado por ambos. Foram identificados 6 operações pela ferramenta e 1 operação pelo P2, não havendo operações identificadas por ambos. Quanto aos relacionamentos, somente 1 foi identificado por ambos, 17 somente pela ferramenta e 6 somente pelo Participante 2.

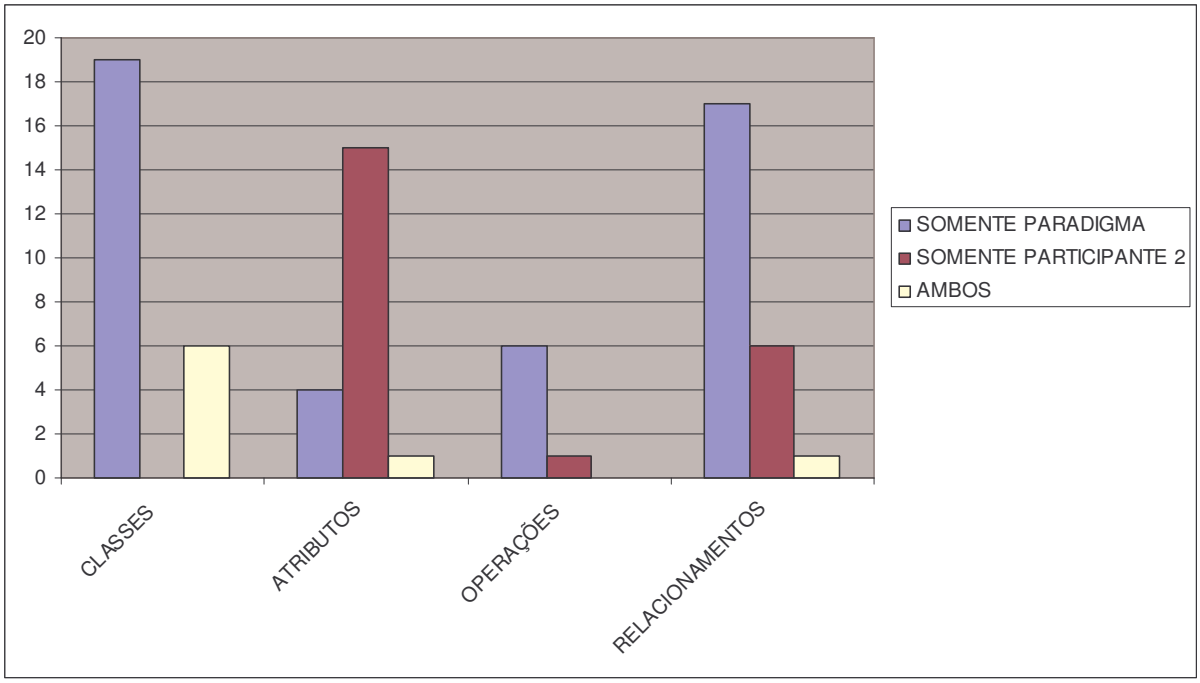


Figura 20 – Gráfico comparativo entre o modelo do P2 e o da ferramenta

Na Figura 21, é apresentado o comparativo entre o modelo gerado pelo P3 com o modelo gerado pela ferramenta PARADIGMA. Foram identificadas 30 classes, onde 16 classes foram identificadas somente pela ferramenta PARADIGMA; 5 classes identificadas somente pelo P3 e 9 classes foram identificadas por ambos. Nesse comparativo, tivemos 64% das classes identificadas pelo P3, contempladas pelo modelo gerado pela ferramenta PARADIGMA. Foram identificados 57 atributos, nos quais 5 atributos foram identificados somente pela ferramenta; 52 atributos foram identificados somente pelo P3 e não houve atributos identificados por ambos. O P3 não identificou as operações, somente a ferramenta identificou 6 operações. Quanto aos relacionamentos, somente 4 foi identificado por ambos, 14 somente pela ferramenta e 13 somente pelo P3.

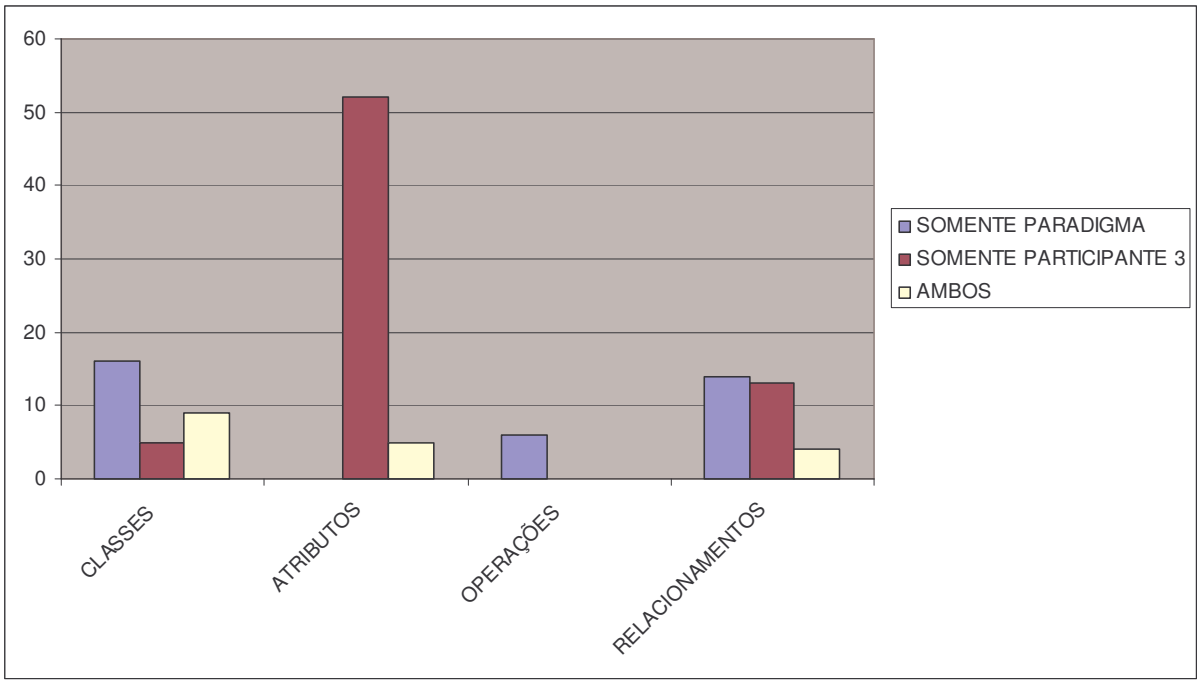


Figura 21 – Gráfico comparativo entre o modelo do P3 e o da ferramenta

Na Figura 22, é apresentado o comparativo entre o modelo gerado pelo P4 com o modelo gerado pela ferramenta PARADIGMA. Foram identificadas 25 classes, onde 21 classes foram identificadas somente pela ferramenta PARADIGMA, 4 classes foram identificadas por ambos. Nesse caso, tivemos 100% das classes identificadas pelo P4, contempladas pelo modelo gerado pela ferramenta PARADIGMA. Foram identificados 13 atributos, onde 8 atributos foram identificados somente pelo P4 e 5 atributos foram identificados por ambos. Foram identificados 6 operações pela ferramenta e 1 operação pelo P4, não havendo operações identificadas por ambos. Quanto aos relacionamentos, 18 deles foram identificados somente pela ferramenta e 4 somente pelo P4, não havendo relacionamentos identificados por ambos.

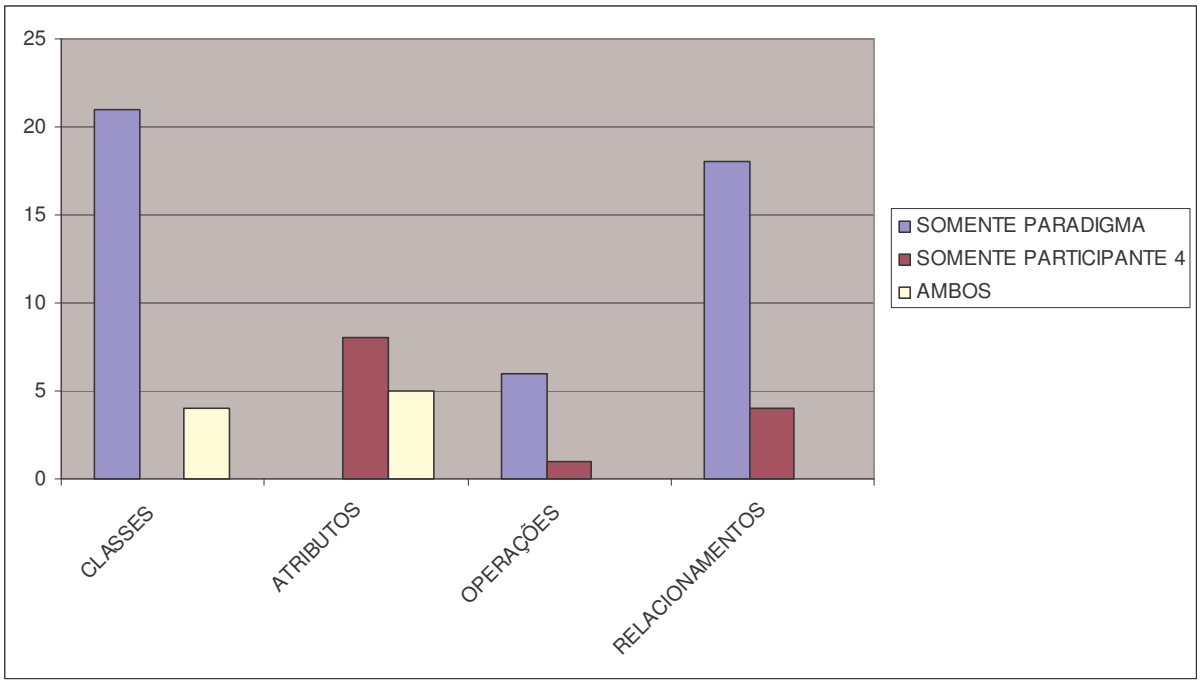


Figura 22 – Gráfico comparativo entre o modelo do P4 e o da ferramenta

5.3.4 TEMPO UTILIZADO PARA GERAR O MODELO DE CLASSES

Os participantes da experimentação da ferramenta PARADIGMA deveriam anotar o tempo necessário para o desenvolvimento das seguintes atividades dentro do processo: gerar o modelo conceitual de classes, sem o auxílio da ferramenta, e ajustar o modelo conceitual de classes gerado automaticamente pela ferramenta para o modelo conceitual de classes gerado pelo próprio participante.

Na Tabela 9 é apresentado o tempo utilizado para a realização das atividades mencionadas acima. Constatamos que a maioria dos participantes realizou os ajustes no modelo gerado pela ferramenta, em menos tempo do que a modelagem feita manualmente. Apenas o Participante 1 realizou o ajuste em mais tempo do que o modelado manualmente, mas ele adicionou a seguinte observação referente a esse aumento de tempo: “demanda de tempo para organizar as classes geradas pela ferramenta, compreender o proposto e determinar o que deveria ser mudado”.

Devemos considerar alguns aspectos quanto ao tempo utilizado para realizar cada atividade. Primeiro, o ajuste do modelo a partir do gerado pela ferramenta, tendo conhecimento de como o modelo deve ficar, facilita a atividade. Segundo, deve ser considerada também a falta de conhecimento referente a

ferramenta PARADIGMA pelos participantes. Eles realizaram as atividades somente consultando o arquivo de ajuda disponibilizado junto com a ferramenta. Esses aspectos citados podem ter influenciado o resultado final dos tempos apresentados na Tabela 9.

Tabela 9 – Tempo utilizado para a modelagem dos diagramas de classes

	Modelagem Manual			Modelagem com auxílio da ferramenta		
	Horário de Início	Horário de Término	Tempo (Minutos)	Horário de Início	Horário de Término	Tempo (Minutos)
Participante 1	08:45	09:10	25	09:30	10:20	50
Participante 2	11:02	11:30	28	11:33	12:00	27
Participante 3	21:30	22:30	60	12:30	12:45	15
Participante 4	20:45	21:05	20	21:40	21:58	18

5.3.5 DEFINIÇÃO DO MODELO CONCEITUAL DE CLASSES PADRÃO

Os modelos conceituais de classes gerados pelos participantes apresentam diferenças significativas entre si. Percebemos a dificuldade que existe para chegar a um modelo conceitual de classes considerado “correto”. Alguns adicionaram detalhes a mais, outros geraram o modelo em um nível conceitual, abstraindo os detalhes. A descrição funcional pode ter influenciado, pois não trazia muitas informações referentes aos atributos e operações, fazendo com que os participantes focassem mais na identificação das classes e dos relacionamentos.

Para realizarmos um comparativo com o modelo conceitual de classes gerado pela ferramenta PARADIGMA, criamos um modelo PADRÃO, construído a partir dos elementos que foram identificados por pelo menos 75% dos participantes. Como a ênfase maior dos participantes foi nas classes, foram levados em consideração as classes e os relacionamentos.

Foram identificadas 27 classes, considerando todos os modelos gerados manualmente pelos participantes e dessas 27 classes, apenas 2 apresentaram ambigüidade em sua identificação. Essas classes são COMPRA e VENDA, onde 50% dos participantes identificaram a classe COMPRA, referindo-se à visão do cliente realizando o ato de comprar CDs na loja e 50% dos participantes identificaram a classe VENDA, referindo-se à visão da loja realizando o ato de vender CDs. Para a definição do modelo conceitual de classes padrão, consideramos a classe VENDA/COMPRA como única e definida como VENDA, totalizando então 26 classes identificadas por todos os participantes. A lista de classes identificadas pelos participantes é apresentada na Tabela 10.

As únicas classes que foram identificadas por todos os participantes foram as classes CLIENTE e LOCACAO. Essas duas classes e as demais que estão em destaque na Tabela 10, e seus respectivos relacionamentos, fazem parte do modelo conceitual de classes padrão.

Tabela 10 – Lista de classes identificadas pelos participantes

Classes	P1	P2	P3	P4
CADASTRO	X			
CDS		X	X	X
CDS_VENDIDOS	X			
CHEQUE	X	X		
CLIENTE	X	X	X	X
CONFERE_CDS	X			
DEVOLUCAO			X	
DUPLICATA			X	
FAIXA			X	
FUNCIONARIO	X	X	X	
ITEM_DEVOLUCAO			X	
ITEM_LOCACAO			X	
ITEM_VENDA			X	
LOCACAO	X	X	X	X
LOJA			X	
PAGAMENTO_COMPRA	X			
PAGAMENTO_LOCACAO	X			
RECEBIMENTO	X			
RECIBO				
RECIBO_COMPRA	X			
RECIBO_LOCACAO	X			
REGISTRO_VENDA	X			
RENOVA_CADASTRO	X			
TIPO_PAGAMENTO			X	
VENDA/COMPRA	X	X	X	X
VISTA	X			

Na Figura 23, pode ser visualizado o gráfico das quantidades de classes que repetiram no modelo de cada participante e a Tabela 9 apresenta os dados que originaram esse gráfico. As classes identificadas pelos participantes P1-P2-P3, P2-P3-P4 e Todos são utilizadas no modelo conceitual de classes padrão.

Na Figura 24, é apresentado o modelo conceitual de classes padrão, gerado a partir dos modelos criados pelos participantes. Consideramos nesse modelo somente as classes, relacionamentos e multiplicidades. Os atributos e operações não foram considerados, porque os modelos dos participantes não dispunham de informações suficientes para a realização das comparações com o modelo de classes gerado pela ferramenta PARADIGMA.

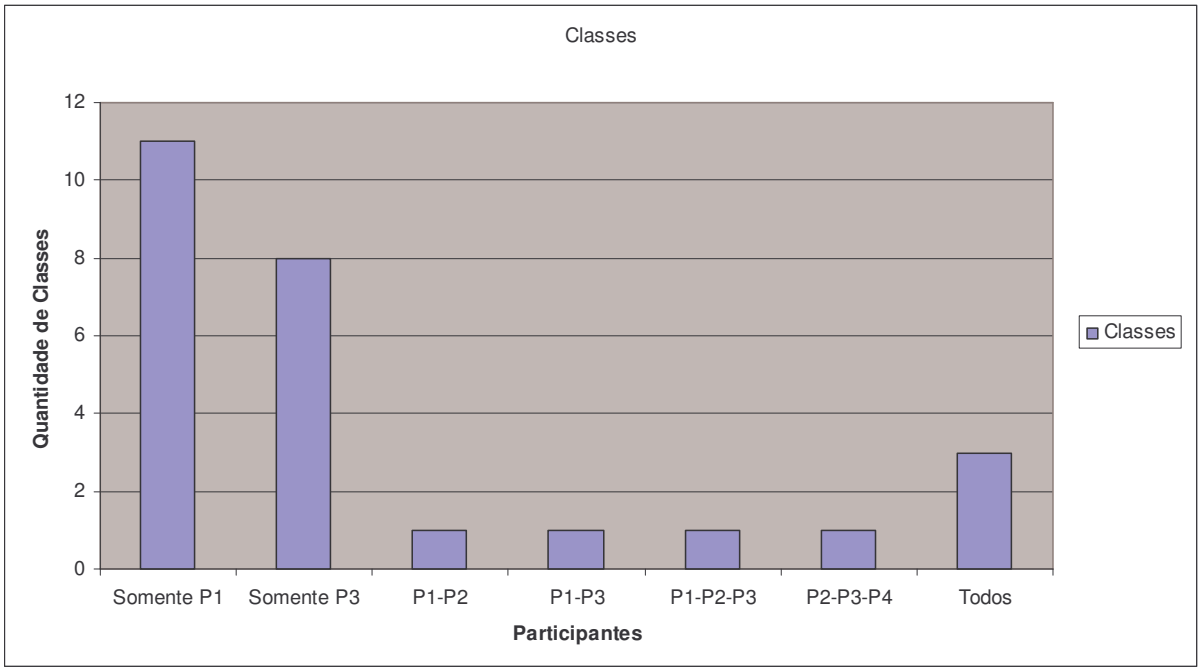


Figura 23 – Gráfico da Junção das classes identificadas pelos participantes

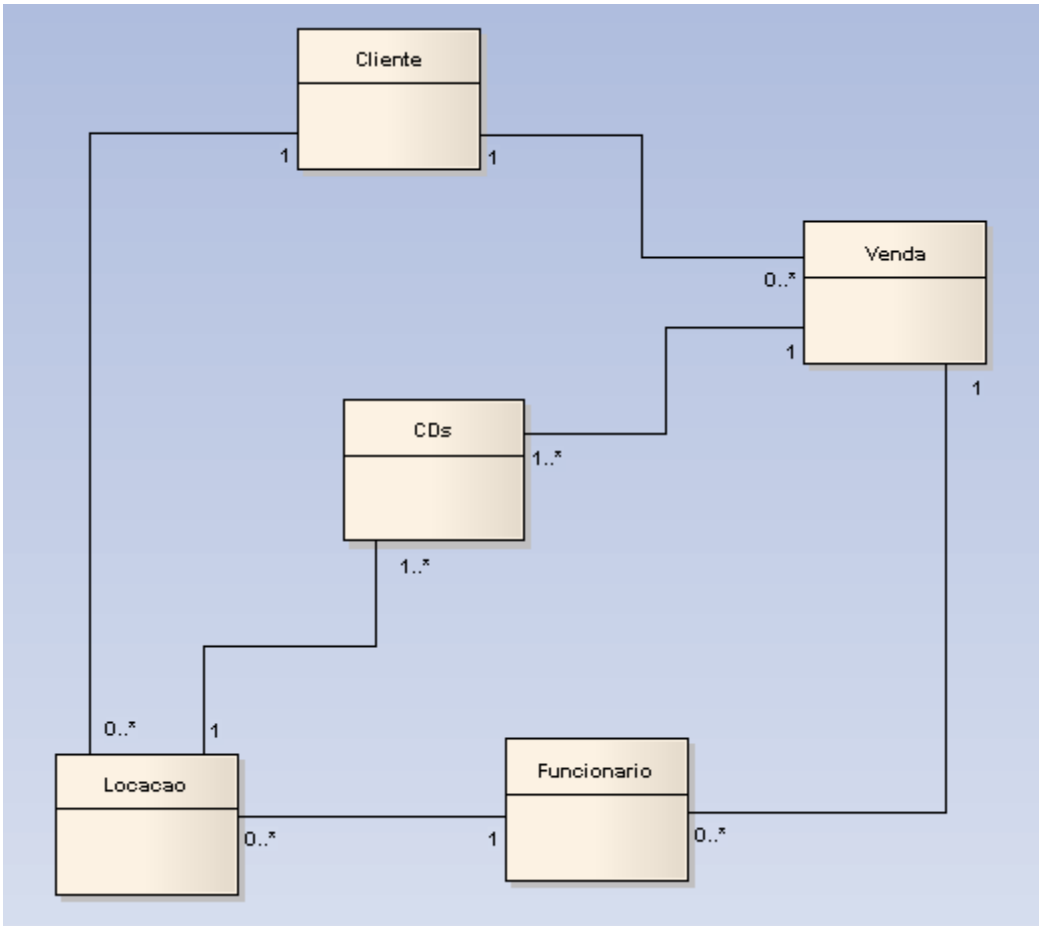


Figura 24 – Modelo Conceitual de Classes Padrão

A partir do modelo de classes apresentado na Figura 24, realizamos o comparativo com o modelo conceitual de classes gerado pela ferramenta

PARADIGMA, da mesma forma que foram comparados os modelos dos participantes individualmente. A Figura 25 traz essa comparação, onde foram identificadas 25 classes, considerando os dois modelos de classes, o padrão e o gerado pela ferramenta PARADIGMA. Constatamos que 100% das classes do modelo conceitual de classes padrão foram identificadas no modelo conceitual de classes gerado pela ferramenta. A ferramenta identificou 20 classes a mais do que as apresentadas no modelo padrão. Quanto aos relacionamentos, foram identificados 23, dos quais 17 somente pela ferramenta PARADIGMA, 5 somente no modelo padrão e 1 relacionamento nos dois modelos.

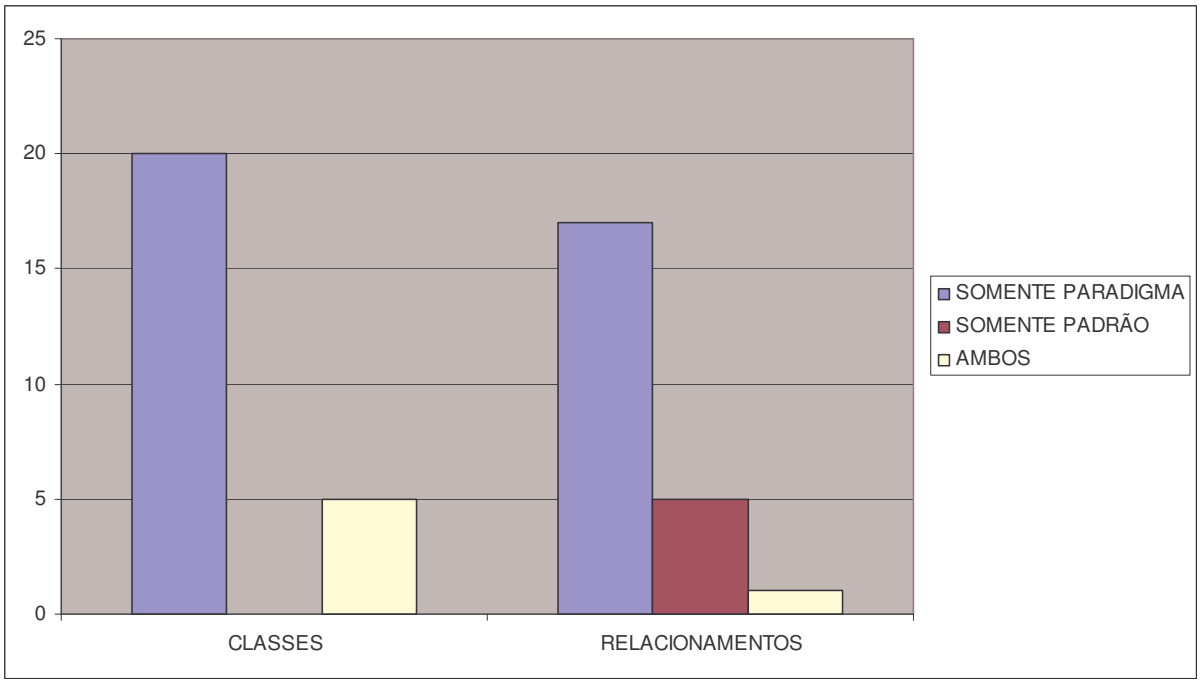


Figura 25 – Gráfico comparativo entre o modelo padrão e o da ferramenta

5.3.6 AVALIAÇÃO DA USABILIDADE DA FERRAMENTA PARADIGMA

Através do cenário de utilização (Apêndice A), os participantes puderam avaliar algumas características da ferramenta PARADIGMA para indicar o grau de concordância quanto à sua facilidade de uso. Essa avaliação foi feita através de um questionário que utiliza a escala de *Likert* de 5 pontos. Para a análise dos resultados foi utilizado uma abordagem quantitativa para mensurar o grau de concordância dos participantes referente à pesquisa realizada. Utilizamos os

seguintes pesos para cada ponto da escala: Concordo Completamente (5), Concordo (4), Indiferente (3), Discordo (2), Discordo Completamente (1).

A Tabela 11 apresenta os resultados do questionário e as respectivas médias ponderadas por questão e a média geral. A média ponderada foi feita utilizando a seguinte fórmula: MP = (Peso x Quantidade de Resposta)/Total de Participantes. Os resultados maiores que 3 são considerados concordantes, os menores que 3 são considerados discordantes e 3 são considerados indiferentes.

Tabela 11 – Questionário de avaliação de usabilidade da ferramenta

	Concordo Completamente	Concordo	Indiferente	Discordo	Discordo Completamente	**Média Ponderada**
Pesos →	**5**	**4**	**3**	**2**	**1**	
1) A ferramenta permite que o engenheiro de requisitos realize as tarefas de forma intuitiva.	1	3	0	0	0	4,25
2) Para utilizar a ferramenta é dispensável a utilização do arquivo de ajuda (AJUDA.PDF).	1	1	0	2	0	3,25
3) O modelo conceitual de classes gerado automaticamente pela ferramenta auxilia o engenheiro de requisitos de forma satisfatória.	0	3	0	1	0	3,50
4) A ferramenta provê recursos necessários para representar satisfatoriamente um modelo conceitual de classes usando a linguagem UML.	3	1	0	0	0	4,75
5) O ajuste do modelo de classes gerado pela ferramenta, para o modelo de classes que o engenheiro de requisitos indica como correto, demanda pouco esforço.	1	1	0	1	1	3,00
Média Geral						**3,75**

A questão 2 ficou bem próxima da indiferença. Os participantes justificaram que necessitaram utilizar o arquivo de ajuda para realizar algumas tarefas, mesmo com a ferramenta apresentando um alto nível de intuitividade. A questão 5 foi avaliada como indiferente. As observações referentes à questão 5 foram as seguintes:

- Uma grande quantidade de elementos identificados pela ferramenta que não foram utilizados no modelo final teve que ser excluídos, demandando um grande esforço;

- O modelo de classes gerado pela ferramenta estava desarranjado e algumas classes e relacionamentos que a ferramenta identificou, mas que não foram utilizados, consumiram certo tempo para serem removidos.

A questão 3 enfrenta o mesmo problema da questão 5, onde a ferramenta apresenta mais elementos que os identificados pelos participantes em seus respectivos modelos.

As questões 1 e 4 foram bem avaliadas, indicando que a ferramenta é intuitiva e permite representar um modelo conceitual de classes usando a linguagem UML de forma satisfatória.

5.4 Análise e Discussão dos Resultados

No cenário de utilização proposto para a experimentação da ferramenta, buscamos a validação da ferramenta PARADIGMA para uso no contexto da Engenharia de Requisitos. Os resultados da experimentação foram de grande valia para descobrirmos alguns desafios pertinentes à utilização do processamento de linguagem natural para a Língua Portuguesa, algumas propostas de trabalhos futuros e algumas melhorias na ferramenta PARADIGMA.

De acordo com Aires (2000), ainda não se atingiu o estado da arte, no que diz respeito à etiquetagem morfossintática para a Língua Portuguesa, fato já atingido para a Língua Inglesa, mas temos vários pesquisadores na área de lingüística dedicando esforços para atingir esse patamar.

Percebeu-se que os padrões lingüísticos auxiliaram, de forma efetiva, na descoberta dos elementos do modelo conceitual de classes a partir de textos em linguagem natural, tornando-se um diferencial frente às ferramentas que se propõem a realizar atividades similares às realizadas pela ferramenta PARADIGMA. Constataram-se, após uma bateria de testes, que a identificação dos elementos do modelo conceitual de classes com a aplicação dos padrões lingüísticos, ocorre com um índice maior de acertos quando o texto em linguagem natural está escrito na voz ativa, com redução da utilização de sujeito oculto nas orações.

Percebeu-se que algumas palavras que não fazem parte do domínio da aplicação são identificadas como classes, atributos ou operações, demandando esforço por parte do engenheiro de requisitos para remoção desses elementos. Nos

trabalhos futuros pode-se avaliar a possibilidade da ferramenta trabalhar em conjunto com uma base de conhecimento, onde pudessem ser catalogadas palavras do domínio da aplicação e/ou algumas palavras padrões que devam ser desconsideradas, mesmo havendo ocorrência no texto em linguagem natural. Outro item importante é a possibilidade de desprezar parte do texto no momento de realizar as identificações dos elementos.

Um fator importante percebido nos resultados da experimentação é a questão da ambigüidade. A ambigüidade pode ocorrer em vários momentos e por várias razões. Ficou evidente a ambigüidade em um momento que não se esperava, na criação do modelo conceitual de classes pelos participantes. Quais elementos devem ser modelados em um modelo conceitual de classes? Depende do que se quer mostrar, e assim, cada participante teve uma interpretação diferente de quais elementos deveriam adicionar nesse modelo. Uns desconsideraram os atributos e operações, outros adicionaram atributos e operações que não apareciam na descrição funcional. Alguns identificaram os atributos e operações, mas deixaram de considerar alguns atributos que estavam evidentes na descrição funcional. Quando simplesmente dizemos modelo conceitual de classes, abrimos margem para algo muito abrangente, no qual cada engenheiro de requisitos pode modelar com mais ou menos detalhes e foi justamente o que ocorreu no processo de experimentação da ferramenta. Mesmo com o número reduzido de participantes conseguimos modelos bem diferentes.

Essa diferença nos modelos criados pelos participantes, humanos, nos mostra o quão desafiador é desenvolver uma ferramenta que gere um modelo conceitual de classes automaticamente e atenda as expectativas de cada engenheiro de requisitos que faça uso dela. Ficou evidente, referente às classes, que a ferramenta conseguiu identificar 100% das classes que compunham o modelo padrão. Nas comparações individuais, obtivemos um bom índice de identificação de classes pela ferramenta, iguais as identificadas pelos participantes. O modelo que apresentou o pior índice foi de 53%, sendo que 50% dos participantes tiveram 100% das suas classes identificadas automaticamente pela ferramenta. O que ocorreu em todos os casos é que a ferramenta apresentou elementos excedentes, os quais tiveram de ser excluídos pelo participante para realizar o ajuste para o modelo de classes definido como “correto” por ele.

Os atributos e operações atingiram um índice baixo de aproveitamento devido aos resultados apresentados pelos participantes. Em alguns momentos foram apresentados atributos e operações que não constavam na descrição funcional, caso impossível da ferramenta identificar. Em outros momentos, atributos apresentados na descrição funcional foram simplesmente ignorados, ou seja, a ferramenta identificou, mas o participante não.

Quanto aos relacionamentos, são necessárias melhorias na especificação dos padrões lingüísticos para identificá-los de forma mais eficiente. Atualmente, o escopo da definição de um relacionamento está restrito a uma frase terminada por ponto ou ponto e vírgula.

Os resultados referentes ao tempo gasto para ajustar o modelo gerado pela ferramenta e o questionário para avaliar a facilidade de uso da ferramenta, foram obtidos através da utilização da ferramenta pelos participantes, sem nenhuma forma de treinamento. Os participantes baixaram a ferramenta em um *link* na internet e realizaram as atividades de acordo com roteiro de utilização e o arquivo de ajuda.

Após a criação do modelo conceitual de classes de forma convencional e o ajuste do modelo gerado automaticamente pela ferramenta, o tempo utilizado para cada uma dessas atividades foi anotado. Percebemos que somente um dos participantes demorou mais tempo no processo de ajuste do modelo gerado pela ferramenta, do que na atividade de criar o modelo da forma convencional. O tempo dos demais participantes se equivaleu nas duas atividades. Quanto ao questionário, o posicionamento, ou o grau de concordância dos participantes em relação às afirmativas apresentadas foi de 3,75. O significado disto é que, em regra geral, os participantes concordaram com as afirmativas que avaliaram a intuitividade da ferramenta, facilidade de uso sem o auxílio da ajuda, auxílio satisfatório ao engenheiro de requisitos, recursos suficientes para gerar o modelo conceitual de classes utilizando a linguagem UML. Esse resultado de 3,75 deve ser visto com base na escala de *Likert* que é composto pelos seguintes pontos: concordo completamente (5), concordo (4). Indiferente (3), discordo (2), discordo completamente (1).

As observações dos participantes foram sempre em relação à grande quantidade de elementos identificados pela ferramenta, principalmente os elementos que estão fora do domínio da aplicação. Devemos ponderar o que é

menos trabalhoso no momento de gerar um modelo conceitual de classes. Identificar e adicionar uma classe manualmente ou eliminar as classes que não fazem parte do contexto? Como no cenário de utilização não se previu uma atividade para realizar essa ponderação, não temos base para afirmar qual leva vantagem em relação à outra.

Os resultados mostraram que a ferramenta PARADIGMA é intuitiva e cumpriu eficazmente seu papel no processo de identificação de classes, auxiliando o engenheiro de requisitos em uma fase fundamental do ciclo de vida do software. A identificação dos relacionamentos necessita de melhorias e os resultados das identificações dos atributos e operações foram comprometidos pelos resultados apresentados pelos participantes.

6 CONCLUSÃO

6.1 CONSIDERAÇÕES INICIAIS

Este trabalho foi norteado pela necessidade, no contexto da Engenharia de Requisitos, de ferramentas que auxiliem os engenheiros de requisitos nas atividades de elicitação, análise, especificação, validação e gerenciamento de requisitos. O engenheiro conta com um conjunto de técnicas que o auxilia em cada atividade do processo de Engenharia de Requisitos, principalmente na atividade de elicitação, onde está focada a ferramenta desenvolvida neste trabalho.

Conforme constatado por (MICH; FRANCH; NOVI INVERARDI, 2003), a maioria dos documentos disponíveis na atividade de elicitação e especificação de requisitos encontra-se em linguagem natural comum, pois as técnicas utilizadas, principalmente na atividade de elicitação, produzem esses documentos em linguagem natural, que é o caso da entrevista, do *JAD*, da etnografia, etc. A partir desses documentos em linguagem natural, o engenheiro de requisitos através de métodos heurísticos, chega a uma solução para o problema, que, no caso em questão, é a especificação dos requisitos. Essa especificação pode ser expressa em linguagem natural, técnicas semiformais ou ainda técnicas formais.

6.2 CONTRIBUIÇÕES

A ferramenta desenvolvida neste trabalho auxilia o engenheiro de requisitos na atividade de especificação de requisitos, utilizando os recursos de PLN, mais especificamente os etiquetadores morfossintáticos. A ferramenta utiliza as marcações adicionadas nos textos em linguagem natural pelos etiquetadores, em conjunto com os padrões lingüísticos pré-definidos, para identificar as classes, atributos, operações, associações e multiplicidades, gerando automaticamente o modelo conceitual de classes da UML. Como o artefato produzido pela ferramenta é o modelo conceitual de classes, fica evidente que o foco está direcionado para a Orientação a Objetos.

A experimentação da ferramenta PARADIGMA foi realizada por professores e profissionais da área de Engenharia de Requisitos, os resultados

mostraram-nos informações relevantes sobre a ferramenta, das quais podemos destacar:

- A ferramenta apresentou um bom desempenho na identificação das classes. Os percentuais de classes identificadas pela ferramenta que coincidiram com as classes criadas pelos participantes variaram entre 53% a 100%. O modelo conceitual de classes padrão, que foi criado a partir dos modelos gerados pelos participantes, teve 100% de suas classes atendidas pela ferramenta PARADIGMA.
- Os atributos que estavam na descrição funcional foram identificados pela ferramenta, mas não foram adicionados automaticamente no modelo, para facilitar o ajuste pelo engenheiro de requisitos, caso a identificação fosse feita de forma equivocada pela ferramenta (os participantes não forneceram informações suficientes para a realização de uma análise mais detalhada deste aspecto).
- Um bom número de operações foi identificado pela ferramenta, ficando disponível para os engenheiros de requisitos vincularem às respectivas classes (os participantes não forneceram informações suficientes para a realização de uma análise mais detalhada deste aspecto).
- O método adotado para identificar as associações foi ineficiente, necessita de melhorias para que se possa aumentar sua eficiência.
- A ferramenta identificou um alto número de elementos que não fazem parte do domínio da aplicação, o que demandou certo esforço dos participantes no processo de exclusão desses elementos.
- O tempo gasto para gerar o modelo conceitual de classes foi praticamente o mesmo, com e sem o auxílio da ferramenta. Devemos considerar o fato de que os participantes não conheciam a ferramenta PARADIGMA, até o momento da experimentação.

Os participantes responderam também um questionário que utilizou a escala de *Likert* com o intuito de avaliar o grau de concordância dos participantes referente à facilidade de uso da ferramenta PARADIGMA. As possibilidades de respostas na escala eram as seguintes: caso a média ficasse abaixo de 3, os participantes estariam discordando em relação a facilidade de uso da ferramenta. Caso a média fosse 3, os participantes estariam indiferentes em relação a facilidade

de uso. Caso a média ficasse acima de 3, os participantes estariam concordando com a facilidade de uso da ferramenta.

O resultado final do questionário ficou em **3,75,** o que demonstrou que os participantes concordaram com a facilidade de uso da ferramenta. O que fez com que o grau de concordância ficasse somente nos 3,75 foi o fato da ferramenta identificar alguns elementos que estão fora do domínio da aplicação, resultando assim em um modelo conceitual de classes um pouco mais complexo, do que se fossem considerados somente os elementos do domínio da aplicação, demandando então tempo para eliminar esses elementos e ajustar o modelo.

Com base nos resultados obtidos, podemos afirmar que a ferramenta PARADIGMA oferece benefícios para os engenheiros de requisitos que atuam com o conceito de orientação a objetos. O modelo conceitual de classes gerado por ela permite que o engenheiro de requisitos visualize de forma gráfica o contexto da aplicação. Mesmo que sejam identificados alguns elementos além do necessário, a ferramenta permite o ajuste do modelo para a realidade desejada pelo engenheiro de requisitos de forma fácil e intuitiva.

6.3 TRABALHOS FUTUROS

Perceberam-se durante o desenvolvimento da ferramenta e através dos resultados obtidos, alguns segmentos que podem ser explorados em trabalhos futuros: explorar os diagramas dinâmicos da UML, pois a maioria das ferramentas está focada nos diagramas estáticos; utilizar uma base de conhecimentos em conjunto com a ferramenta para facilitar a identificação das palavras que fazem parte do domínio da aplicação e também a implementação de uma *stop list*, ou seja, termos do domínio da aplicação ou da língua em geral que não deve ser considerado no modelo conceitual de classes; buscar soluções para aumento de eficiência na identificação dos relacionamentos, permitindo assim a identificação dos seguintes tipos: generalização, agregação e composição; interagir com ferramentas CASE que atendem às demais fases do ciclo de vida do *software.*

REFERÊNCIAS BIBLIOGRÁFICAS

AIRES, Rachel V.X. **Implementação, adaptação, combinação e avaliação de etiquetadores para o português do Brasil**. São Carlos: USP, 2000. 154 f. Dissertação de mestrado – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, São Carlos, 2000.

ALLEN, James, **Natural Language Understanding**. The Benjamin/Cummings Publishing Company Inc, 1995.

BERRY, D.M.; KAMSTIES, E. Ambiguity in Requirements Specification In: LEITE, J. C. S. P.; DOORN, J. H. **Perspectives on Software Requirements**, Series: The Springer International Series in Engineering and Computer Science , Vol. 753, Kluwer Academic Publishers, 2004, p. 7-44.

BIRD, S., KLEIN, E., and LOPER, E., **Natural Language Processing in Python**, disponível em: <http://nltk.org> Acessado em: Outubro de 2007.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., **UML: Guia do Usuário**, Campus, 2000.

BROOKS, F., **No Silver Bullet: Essence and Accidents of Software Engineering**, Computer, Apr., p. 10-19, 1987.

DAVIS, A.M., HICKEY, A.M. **Elicitation Technique Selection: How Do Experts Do It**, Proceedings of the 11th International Requirements Engineering Conference, IEEE Computer Society Press, 2003.

HARMAIN, H.M., GAIZAUSKAS R. **CM-Builder: An Automated NL-based CASE Tool**, In Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE'2000), 2000, p. 45-53.

IEEE-SA STANDARDS BOARDS IEEE Std 830-1998:**IEEE Recommended Practice For Software Requirements Specifications**, jun. 1998.

JIANG, L., EBERLEIN, A., FAR, B.H, **A Methodology for Requirements Engineering Process Development**, In Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-based Systems (ICBS'04), Brno, Czech Republic, 2004.

LEITE, J.C.S.P., Gerenciando a Qualidade de Software com Base em Requisitos In: ROCHA, MALDONADO, WEBER, **Qualidade de Software: Teoria e Prática,** São Paulo: Prentice-Hall, 2001.

KOTONYA, G.; SOMMERVILLE, I. **Requirements Engineering: process and techniques,** New York: John Wiley & Sons Ltda, 1998.

LAMSWEERDE, A. **Requirements Engineering in the Year 00: A Research Perspective**. In: International Conference on Software Engineering, 22, jun. 2000, Limerick, Ireland. Proceedings… ACM, p. 5-19, 2000.

LI, K., DEWAR, R.G., POOLEY, R.J. **Requirements capture in natural language problem statements**, Technical Report HW-MACS-TR-0023, Heriot-Watt University Edinburgh, Scotland, UK, 2004.

LI, K., DEWAR, R.G., POOLEY, R.J. **Object-oriented Analysis Using Natural Language Processing**, Technical Report HW-MACS-TR-0033, Heriot-Watt University Edinburgh, Scotland, UK, 2005.

LIU, D., SUBRAMANIAM, K., EBERLEIN, A., FAR, B.H. **Natural Language Requirements Analysis and Class Model Generation Using UCDA**, Springer, 2004.

MARTINS, LUIZ E.G.. **Uma metodologia de elicitação de requisitos de software baseada na teoria da atividade**. Campinas: UNICAMP, 2001. 182 f. Tese de doutorado – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2001.

MICH, L., FRANCH, M. and NOVI INVERARDI, P. **Requirements Analysis using Linguistic Tools: Results of an On-line Survey**. Technical Report 66, Department of Computer and Management Sciences, 2003.

NAUR, P., RANDELL, B. **Software Engineering: Report on a Conference Sponsored by the NATO Science Commission** Garmish, Germany, 7-11 Oct. 1968. Scientific Affairs Division, NATO, Brussels, Jan. 1969.

OVERMYER, S.; LAVOIE, B.; and RAMBOW, O. **Conceptual Modeling through Linguistic Analysis Using LIDA**. In Proceedings of 23rd International Conference on Software Engineering (ICSE 2001), Toronto, Canada, 2001.

PFLEEGER, S.L. **Software Engineering: theory and practice**. Prentice Hall, 1998.

PIATTINI, M. G., CALVO-MANZANO, J. A., CERVERA, J. y FERNÁNDEZ, L. **Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión**. ra–ma, 1996 apud TORO, A. D; JIMENEZ, B. B; Metodología para la Elicitación de Requisitos de Sistemas de Software. Informe Técnico LSI-2000-10. Facultad de Informática y Estadística Universidad de Sevilla, Outubro, 2000.

RATNAPARKHI, A., **A Maximum Entropy Part-Of-Speech Tagger** Proceedings of the Empirical Methods in Natural Language Processing Conference, University of Pennsylvania, 1996.

ROBERTSON, S.; ROBERTSON, J. **Mastering the Requirements Process**. Addison Wesley, 1999.

ROBERTSON, S., **Requirements - Learning From Others Discipline**, IEEE Computer Society, 2005.

SCHWARTZ, J.I. **Construction of Software, Problems and Practicalities**, in Pratical Strategies for Developing Large Software Systems, E. Horowitz, ed., Addison-Wesley, Reading, Mass, 1975.

SOMMERVILLE, I., "Software Engineering", 6. ed., Pearson Education Limited, 2001.

THAYER, R.H., DORFMAN, M., **Software Requirements Engineering**, 2th ed., IEEE Computer Society Press, Los Alamitos, Califórnia, 1997.

TORO, A. D; JIMENEZ, B. B; ***Metodología para la Elicitación de Requisitos de Sistemas de Software.*** Informe Técnico LSI-2000-10. Facultad de Informática y Estadística Universidad de Sevilla, Outubro, 2000.

UNIVERSIDADE METODISTA DE PIRACICABA
FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

APÊNDICE A - CENÁRIO DE UTILIZAÇÃO

PIRACICABA, SP
2007

Sumário

1 INTRODUÇÃO

O objetivo desse cenário de utilização da ferramenta PARADIGMA é avaliar a ferramenta através do uso da mesma por professores e profissionais da área de Engenharia de Software, mais especificamente, do segmento da Engenharia de Requisitos.

Após a realização do cenário de utilização, iremos desenvolver as seguintes atividades:

- Comparar o modelo gerado pelo engenheiro de requisitos com o modelo gerado pela ferramenta, a partir de uma mesma descrição funcional.
- Comparar o tempo e/ou dificuldades para se chegar a um modelo de classes conceitual, considerado correto pelo engenheiro de requisitos, feito manualmente e gerado automaticamente pela ferramenta, a partir da mesma descrição funcional.
- Avaliar a relevância da ferramenta para o engenheiro de requisitos na atividade de modelar um modelo conceitual de classes, a partir de descrições funcionais.

2 CENÁRIO DE UTILIZAÇÃO

A atividade proposta para a avaliação da ferramenta é bem sucinta. A seguir, é apresentada à descrição funcional que servirá de base para a realização da experimentação da ferramenta.

Com base nessa descrição funcional, o engenheiro de requisitos deverá realizar as atividades descritas a partir do item 2.2.

2.1 DESCRIÇÃO FUNCIONAL – SISTEMA PARA UMA LOJA DE CDS.

Uma loja de CDs possui CDs para venda e para locação. Um cliente pode comprar ou locar uma quantidade ilimitada de CDs. Para locar, é obrigatório que o cliente esteja cadastrado na loja, ou seja, tenha preenchido uma ficha de cadastro que deva ser renovada a cada 06 meses. Na ficha de cadastro, deve constar: nome do cliente, endereço, data de cadastro, telefone e e-mail.

As vendas de CDs podem ser efetuadas à vista com 10 % de desconto, ou sem desconto, através de cheque pré-datado, descontado 30 dias após a compra.

As locações somente podem ser pagas à vista, na devolução dos CDs, que deverá acontecer 02 dias após a locação. O valor da locação do CD é R$ 1,00.

A loja possui um funcionário, cuja função é atender os clientes durante a venda e locação dos CDs e suas principais tarefas são:

- Receber e conferir o pagamento efetuado pelos clientes;
- Emitir recibo de venda e locação (este último em duplicata) ;
- Anotar em uma caderneta o valor de cada venda, assim também como os CDs vendidos;
- Conferir o estado dos CDs devolvidos.

2.2 GERAÇÃO DO MODELO DE CLASSES PELO ENGENHEIRO DE REQUISITOS MANUALMENTE

A partir da descrição funcional descrita acima, o engenheiro de requisitos deverá modelar o modelo conceitual de classes utilizando sua ferramenta de modelagem de preferência (JUDE, EA, Rational Rose...) ou manualmente, como preferir. Se possível, essa atividade deve ser desenvolvida sem interrupção. Caso não seja possível anotar os horários de início e término de cada etapa do processo.

Exemplo:

Modelo – engenheiro de requisitos manualmente	
Horário de Início	Horário de Término
08:00	09:00
11:30	12:00

Nesse ponto, teremos o modelo conceitual de classes confeccionado pelo engenheiro de requisitos.

2.3 GERAÇÃO DO MODELO DE CLASSES COM O AUXÍLIO DA FERRAMENTA

Realizar a atividade anterior utilizando a ferramenta PARADIGMA. Para a utilização da ferramenta, serão necessários alguns procedimentos básicos:

- **Instalar o *Java Runtime Environment* [máquina virtual do Java - JVM]**: caso o JVM não esteja instalado no computador onde será utilizada a ferramenta, poderá ser instalado utilizando o arquivo ***JRE-6U1-WINDOWS-I586-P-S.EXE*** que encontra-se na pasta \ER-FERRAMENTA do CD ou baixar do endereço http://www.unasp-sp.edu.br/~wilson/er/***JRE-6U1-WINDOWS-I586-P-S.EXE*** .

- Copiar a pasta **ER-FERRAMENTA** do CD para C:\ ou baixar no endereço http://www.unasp-sp.edu.br/~wilson/er/er-ferramenta.zip e descompactar em C:\.

O objetivo, nesse tópico 2.3, é chegar ao mesmo modelo conceitual de classes gerado no tópico 2.2, mas agora, com o auxilio da ferramenta. O engenheiro de requisitos deverá realizar os ajustes necessários no modelo gerado automaticamente pela ferramenta.

O arquivo de ajuda da ferramenta está na seguinte pasta **C:\ER-FERRAMENTA\AJUDA.PDF**.

Para entrar na ferramenta, execute o arquivo **C:\ER-FERRAMENTA\PARADIGMA.EXE.**

IMPORTANTE: caso tenha alguma dificuldade em baixar os arquivos nos endereços especificados e/ou em realizar a instalação da ferramenta PARADIGMA, poderá entrar em contato pelo e-mail:
wilson.carlos@unasp.edu.br ou **wilsonwcs@terra.com.br.**

2.3.1 PASSOS PARA GERAÇÃO AUTOMÁTICA DO MODELO DE CLASSES

- Clique no botão **[Captura Classes/Atributo/Operações]** no lado direito da janela principal.
- Selecione o arquivo [**C:\ER-FERRAMENTA\CENARIO.TXT]** e clique em [**Abrir].**
- O modelo será gerado automaticamente. O ajuste das classes na tela poderá ser feito utilizando o recurso de arrastar e soltar. Os atributos e operações podem ser arrastados para suas classes respectivas no modelo.
- Para adicionar generalização, multiplicidades, classes... consulte **C:\ER-FERRAMENTA\AJUDA.PDF**
- Informar horário de início e término dessa atividade, similar ao que foi feito no tópico 2.1.

  Exemplo:

Modelo – Gerado automaticamente + Interferência do Engenheiro de Requisitos	
Horário de Início	Horário de Término
08:00	09:00
11:30	12:00

- Depois de realizado os ajustes devidos no modelo conceitual de classes gerado pela ferramenta para ficar similar ao gerado na

opção 2.2, o modelo deverá ser salvo utilizando a seguinte opção [**Arquivo**/**Salvar Diagrama...**] no menu principal.

- Sair da Ferramenta **PARADIGMA.**

2.4 QUESTIONÁRIO - AVALIAÇÃO DE UTILIZAÇÃO DA FERRAMENTA

1) A ferramenta permite que o engenheiro de requisitos realize as tarefas de forma intuitiva.

a) Concordo Completamente
b) Concordo
c) Indiferente
d) Discordo
e) Discordo Completamente

Observação:

_____

_____

_____

2) Para utilizar a ferramenta é dispensável a utilização do arquivo de ajuda (AJUDA.PDF).

a) Concordo Completamente
b) Concordo
c) Indiferente
d) Discordo
e) Discordo Completamente

Observação:

_____

_____

_____

3) O modelo conceitual de classes, gerado automaticamente pela ferramenta, auxilia o engenheiro de requisitos de forma satisfatória.

a) Concordo Completamente
b) Concordo

c) Indiferente
d) Discordo
e) Discordo Completamente

Observação:

_____

_____

_____

4) A ferramenta provê recursos necessários para representar satisfatoriamente um modelo conceitual de classes usando a linguagem UML.

a) Concordo Completamente
b) Concordo
c) Indiferente
d) Discordo
e) Discordo Completamente

Observação:

_____

_____

_____

5) O ajuste do modelo conceitual de classes gerado pela ferramenta, para o modelo conceitual de classes que o engenheiro de requisitos indica como correto, demanda pouco esforço.

a) Concordo Completamente
b) Concordo
c) Indiferente
d) Discordo
e) Discordo Completamente

Observação:

_____

_____

_____

2.5 COMENTÁRIOS/SUGESTÕES

_____
_____
_____
_____

2.6 RETORNO DO CENÁRIO DE UTILIZAÇÃO

Enviar email para os seguintes endereços:

- wilson.carlos@unasp.edu.br
- wilsonwcs@terra.com.br
- lgmartin@unimep.br

Informações que deverão constar no e-mail

- Arquivo no formato *.JPG ou *.BMP ou *.PDF do modelo gerado pelo engenheiro de requisitos no tópico 2.2;
- Tabela de horários do tópico 2.2;
- Tabela de horários do tópico 2.3;
- Anexar o arquivo C:\ER-FERRAMENTA\PARADIGMA.MDB;
- Resposta do questionário do tópico 2.3 (pode ser no corpo do e-mail, exemplo: 1-A, 2-B, 3-C...);
- Comentários/sugestões do tópico 2.5 (pode ser no corpo do e-mail).

Desde já, agradecemos a sua participação e colaboração neste projeto. Manteremos todos os participantes informados sobre os resultados desse projeto.

Apêndice B - Documento de Ajuda <Paradigma>

Versão 1.0

Piracicaba – SP, 19/11/2007.

INTRODUÇÃO

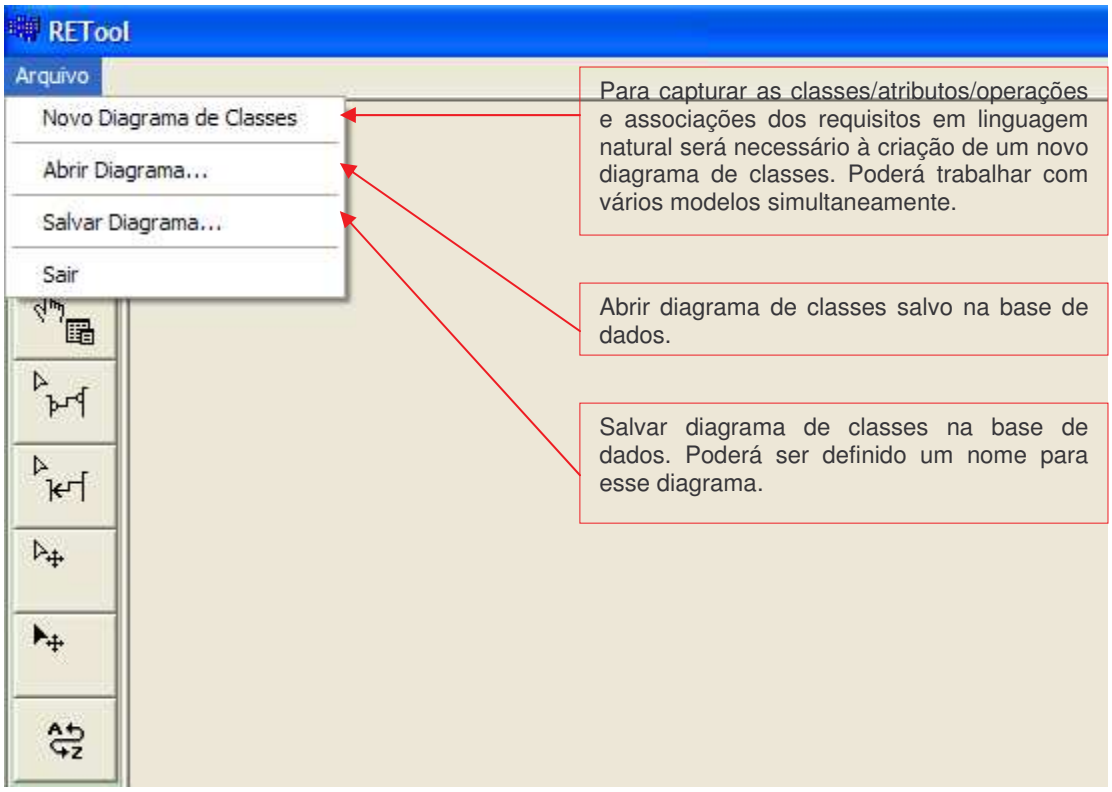
O objetivo deste documento é apresentar as funcionalidades principais da ferramenta PARADIGMA, desenvolvida no projeto de Mestrado em Ciência da Computação, na linha de pesquisa de Engenharia de Requisitos.

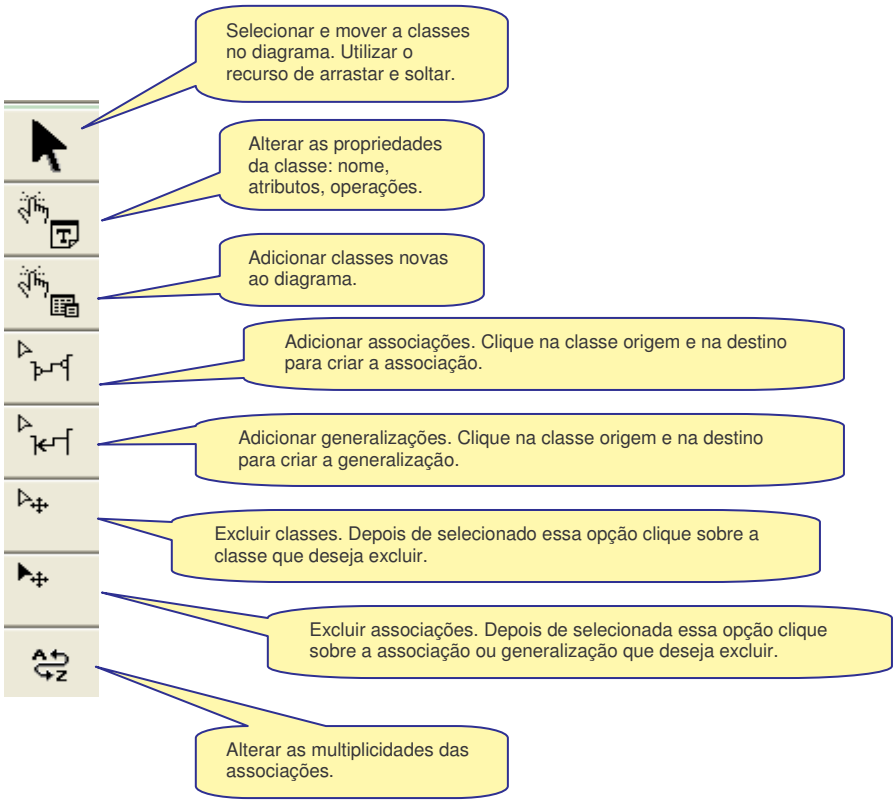
A ferramenta foi desenvolvida com uma *interface* simples e enxuta para facilitar a utilização pelo engenheiro de requisitos. Ela permite a geração automática do modelo conceitual de classes, a partir de requisitos descritos em linguagem natural, e também a interação do engenheiro de requisitos com o modelo para realizar alterações necessárias ao modelo.

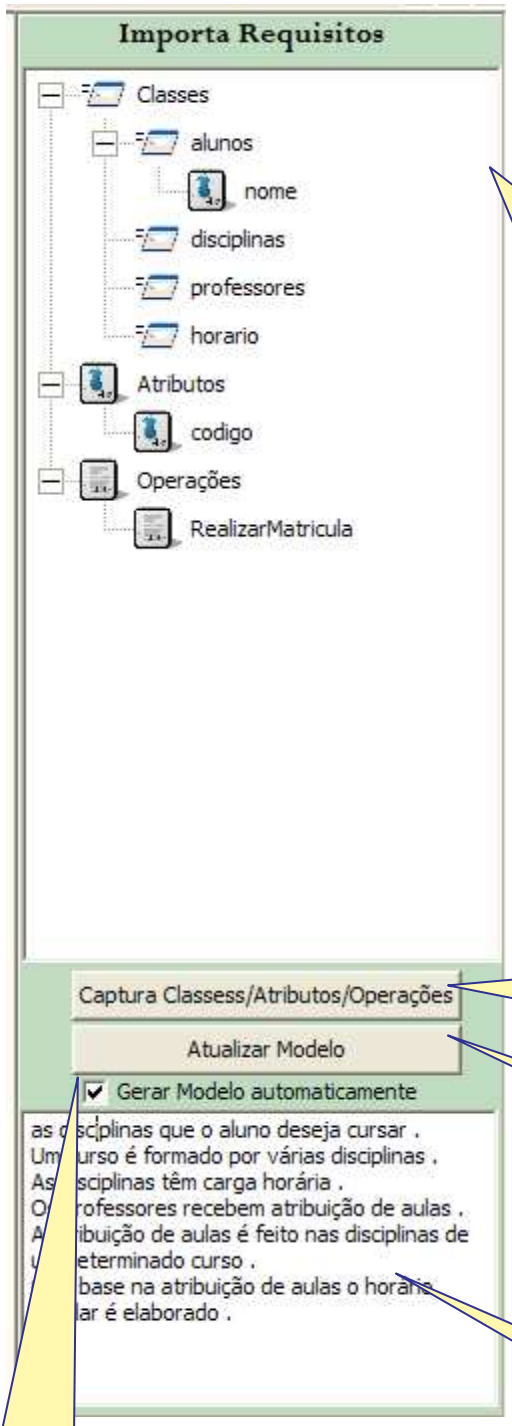
No próximo capítulo, serão apresentadas as telas da ferramenta e suas respectivas aplicações.

TELAS DA FERRAMENTA

As telas e as explicações referentes às suas funcionalidades são apresentadas de forma bem simples para facilitar a rápida leitura do documento.







Lista de candidatas a classes, atributos e operações localizadas a partir dos requisitos em linguagem natural.

- Para alterar uma palavra que está classificada como atributo para classe, deverá clicar e arrastar a palavra em cima da pasta CLASSES.
- Para associar um atributo a uma determinada classe, clique e arraste o atributo para cima da classe desejada no TREEVIEW ou no DIAGRAMA.
- A tecla <DEL> exclui itens da lista de classes, atributos e operações.
- Para alterar a descrição de uma classe, atributo ou operação deverá dar duplo clique em cima da palavra e modificar a descrição.

Captura as palavras de um arquivo formato (*.TXT) e adiciona na lista acima, levando em consideração os padrões lingüísticos definidos na ferramenta para fazer a pré-classificação das palavras em classe, atributo e operação.

Atualiza o modelo manualmente, caso a opção gerar modelo automaticamente esteja desmarcada.

Conteúdo do arquivo utilizado para realizar a captura das classes, atributos e operações.

Quando selecionado, o modelo é gerado automaticamente, sem a interferência do engenheiro de requisitos.

Dica 01: Na exclusão de associações, a associação que será excluída deverá estar em primeiro plano. Caso tenha dificuldade em excluir uma determinada associação, selecione uma das classes envolvidas na associação e mude-a de posição no diagrama para que a associação fique em primeiro plano. Logo após esse procedimento, realize a exclusão da associação.

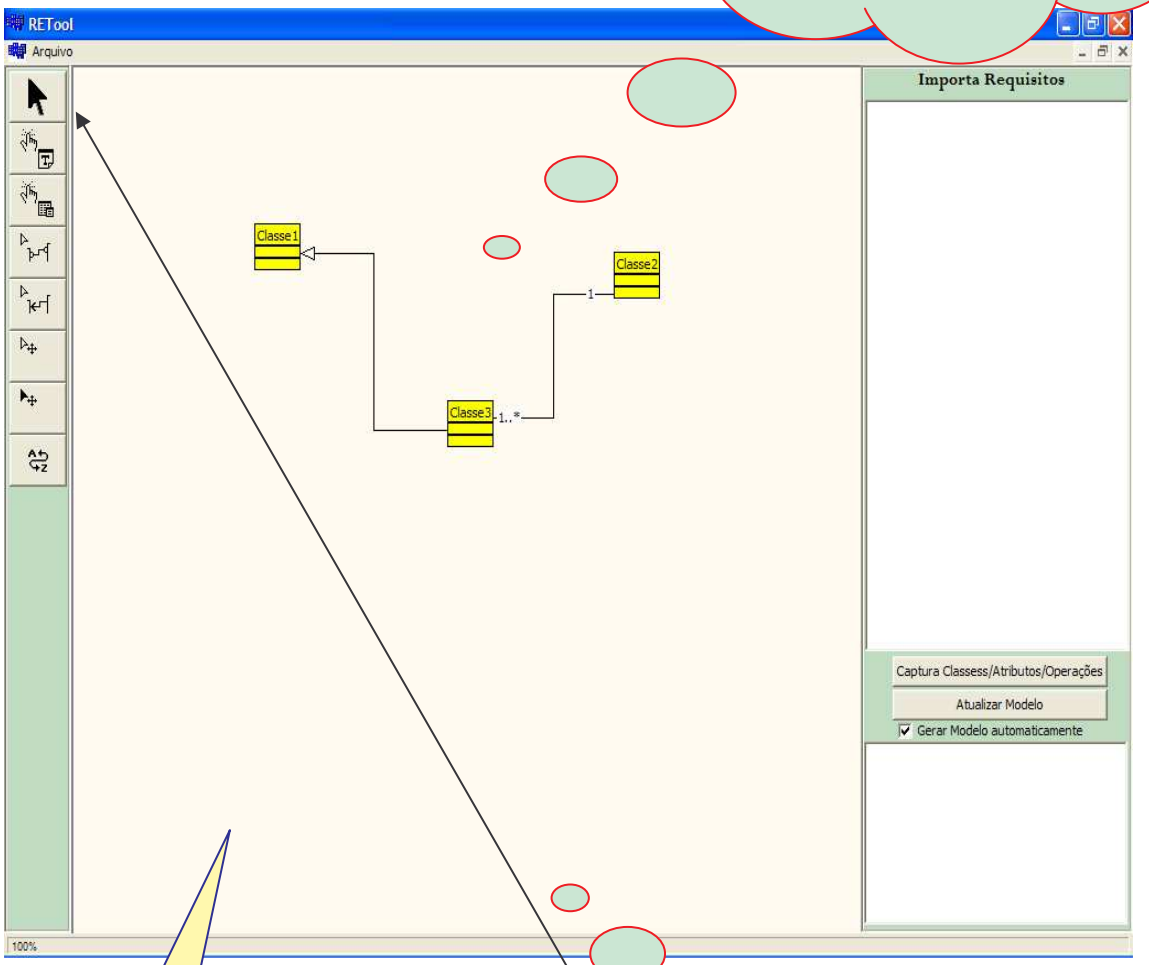
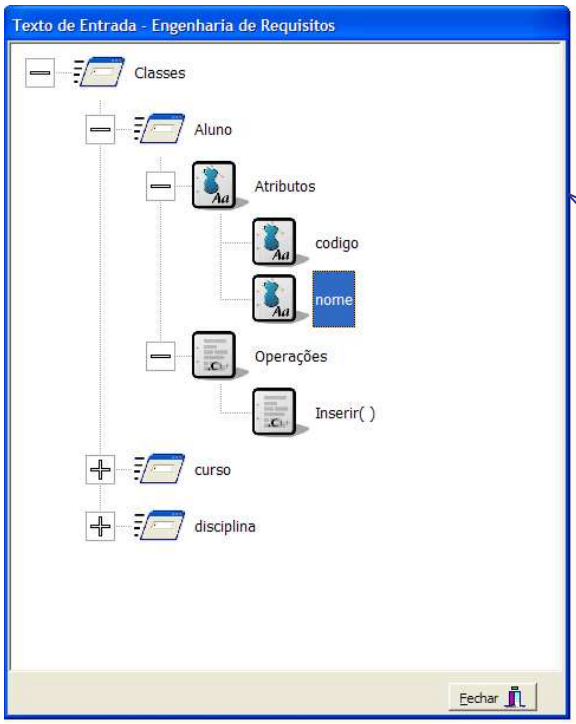


Diagrama de classes → área de trabalho.

- Para mover as classes e reorganizar as associações utilize o recurso de arrastar e soltar.

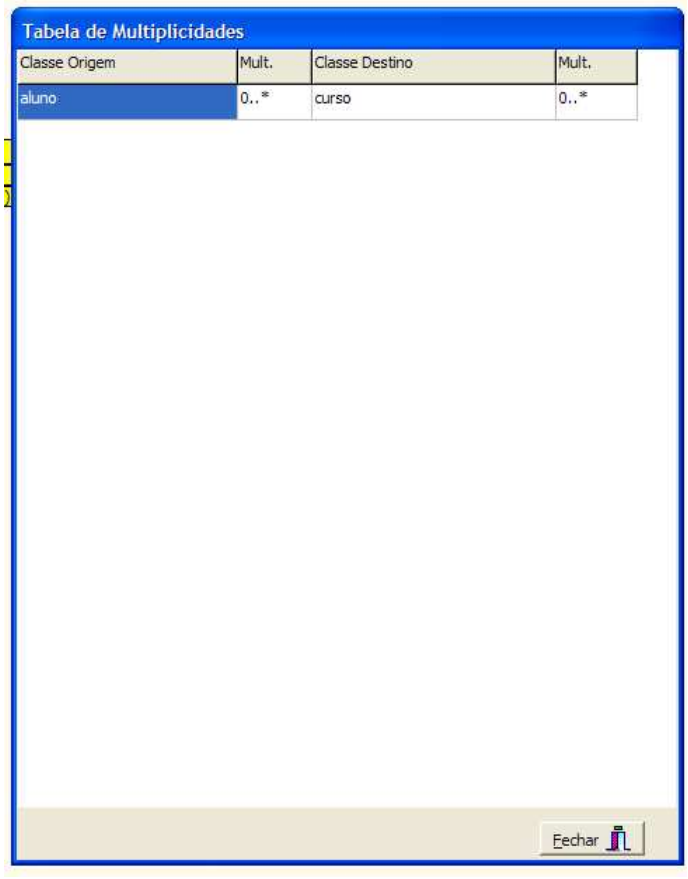
Dica 02: Para selecionar uma classe para arrastar/excluir, as associações que estiverem em cima da respectiva classe deverão ser colocadas em segundo plano. **Procedimento:** Clique na opção SELECIONAR e clique sobre a classe, caso tenha alguma associação sobre ela será exibido a descrição da associação e a associação será colocada em segundo plano. Repita esse procedimento até que a classe fique liberada para execução da operação desejada referente à classe.



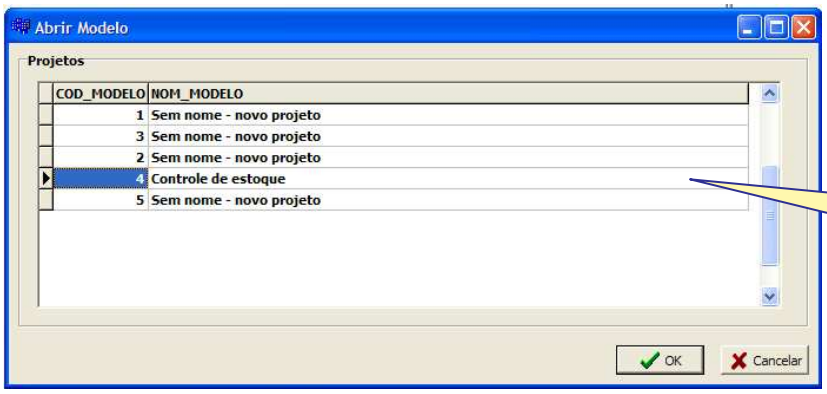
Tela utilizada para alterar o nome da classe, adicionar e excluir atributos manualmente, excluir atributos, adicionar e excluir operações manualmente.

- Teclas

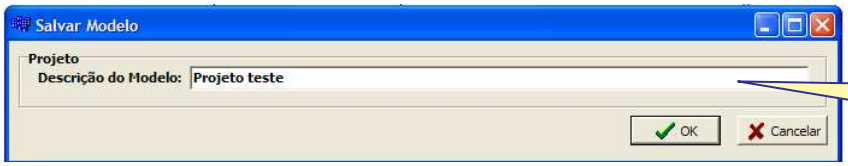
<INSERT> → Insere novo item
<DELETE> → Exclui item
<Duplo Clique> → Altera descrição



Define a multiplicidade de uma determinada associação entre duas classes.



Selecione o modelo que deseja abrir e clique no botão OK



Digite a descrição do modelo que você deseja salvar e clique no botão OK

GLOSSÁRIO

Elicitação: o termo elicitação vem da palavra inglesa *elicitation*, que significa descobrir, desvendar algo obscuro. Embora o termo elicitação não exista no vocabulário da Língua Portuguesa, ele vem sendo aceito e utilizado na comunidade de Engenharia de Requisitos como uma "tradução" do termo *elicitation*.

Corpus: à princípio, qualquer coleção de mais de um texto pode ser chamada de corpus. Mas quando este termo é utilizado em lingüística moderna, tende a ser utilizado com mais freqüência como uma coleção de textos representativos da Língua, que tem tamanho finito, está disponibilizada em formato eletrônico e é um padrão de referência.

Máxima Entropia: entropia é uma medida matemática de ignorância, é inversamente proporcional à informação, isto é, quando o número de informações diminui, a entropia aumenta (alta entropia é sinônimo de alta ignorância). Máxima entropia pressupõe que todas as probabilidades são iguais e independentes umas das outras. Mínima entropia existe quando se espera encontrar uma única possibilidade.

***Stakeholder*:** o termo *stakeholder* foi criado para designar todas as pessoas, instituições ou empresas que, de alguma maneira, são influenciadas pelas ações de uma organização.