COPPE/UFRJ

METODOLOGIA PARA CONSTRUÇÃO DE MODELOS CONCEITUAIS PARA APLICAÇÃO MULTIRRELACIONAL COM AUXÍLIO DE ONTOLOGIA

Silvio Bortoleto

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientador: Nelson Francisco Favilla Ebecken

Rio de Janeiro
Novembro de 2010

METODOLOGIA PARA CONSTRUÇÃO DE MODELOS CONCEITUAIS PARA APLICAÇÃO MULTIRRELACIONAL COM AUXÍLIO DE ONTOLOGIA

Silvio Bortoleto

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE), DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:

_____
Prof. Nelson Francisco Favilla Ebecken, D.Sc.

_____
Profª. Beatriz de Souza Leite Pires de Lima, D.Sc.

_____
Prof. Elton Fernandes, D.Sc.

_____
Prof. Gilberto Carvalho Pereira, D.Sc.

_____
Profª. Fernanda Araújo Baião, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
NOVEMBRO DE 2010

Bortoleto, Silvio

Metodologia Para Construção de Modelos Conceituais Para Aplicação Multirrelacional Com Auxílio de Ontologia/ Silvio Bortoleto. – Rio de Janeiro: UFRJ/COPPE, 2010.

XXII, 143 p.: il.; 29,7 cm.

Orientador: Nelson Francisco Favilla Ebecken

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Civil, 2010.

Referências Bibliográficas: p. 130-143.

1. Modelos Conceituais. 2. Ontologia. 3. Data Mining. I. Ebecken, Nelson Francisco Favilla. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

DEDICATÓRIA

À minha esposa Kátia e à minha filha Sophia, pela compreensão da importância deste trabalho para a minha realização pessoal e, principalmente, pelo apoio incondicional dedicado durante os momentos mais difíceis.

Agradecimentos

Agradeço a Deus. E agradeço também:

- Ao professor Nelson, pela extrema confiança e apoio neste trabalho. A paciência e orientações foram determinantes para a realização da tese. Sua disponibilidade e amizade foram fatores de motivação.

- À minha mãe Clarice, pelo eterno incentivo aos meus estudos. À minha irmã Sonia e ao Salvageto, pelo apoio recebido durante a realização deste trabalho.

- Ao meu amigo Gustavo Lugo, pelo interesse demonstrado neste trabalho e por suas contribuições nas várias discussões técnicas que tivemos.

- A todos os amigos que me incentivaram na realização deste trabalho.

- Aos profissionais do Hospital onde foram realizadas as pesquisas, pela atenção e pelo carinho.

- Aos professores da COPPE, que muito contribuíram para a minha formação acadêmica.

- Aos professores e professoras que dedicaram o seu precioso tempo para me ensinar.

- Ao amigo João Paulo Bossoni, pelas contribuições e discussões técnicas.

Agradeço especialmente à Egna, pela sua dedicação e ajuda em todos os momentos durante o trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

METODOLOGIA PARA CONSTRUÇÃO DE MODELOS CONCEITUAIS PARA APLICAÇÃO MULTIRRELACIONAL COM AUXÍLIO DE ONTOLOGIA

Silvio Bortoleto

Novembro/2010

Orientador: Nelson Francisco Favilla Ebecken

Programa: Engenharia Civil

Todas as áreas de conhecimento oferecem um campo vasto para aplicação de ontologias, metodologias e técnicas ligadas a banco de dados, como modelagem, data warehouse e mineração de dados. A área da saúde oferece possibilidades de aplicações destas técnicas devido a complexidades dos processos e o grande volume de armazenamento de seus dados em uso pelos sistemas de informação. Este trabalho apresenta uma metodologia para modelos conceituais com auxilio de ontologia, a partir de um modelo de dados de um hospital, com a finalidade de maior compreensão sobre o modelo de dados e processos e o suporte de mineração de dados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

METHODOLOGY FOR BUILDING CONCEPTUAL MODELS FOR IMPLEMENTING MULTI-RELATIONAL WITH HELP ONTOLOGIES

Silvio Bortoleto

November/2010

Advisor: Nelson Francisco Favilla Ebecken

Department: Civil Engineering

All knowledge areas offer considerable scope for application of ontologies, methodologies and techniques associated with the database as modeling, data warehouse and data mining. The health area offers possibilities of applications of these techniques because the complexities of processes and the large storage volumes of data in their use of information systems. This paper presents a methodology for conceptual models with the aid of ontology from a data model of a hospital, with the aim of better understanding of the data model and processes and support data mining.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Todas as áreas do conhecimento oferecem um campo vasto para aplicação de metodologias e ferramentas que utilizam ontologias e que podem auxiliar técnicas ligadas a banco de dados, como modelagem, data warehouse e mineração de dados. Um dos campos que mais abre espaço é o da mineração de dados, que auxilia os profissionais da área na descoberta de padrões. A saúde é uma área que oferece possibilidades de aplicações de ontologias para descrever e entender a complexidade dos processos envolvidos nos sistemas informacionais utilizados nesta área, bem como o armazenamento de seus dados. A tarefa não se resume apenas à aplicação de algoritmos sobre a massa de dados, mas também na elaboração de um conhecimento que denote uma representação condizente com a realidade para a procura de resultados que sejam inteligíveis e que as descobertas encontrem relação com o modelo descrito (SARKER e NEWTON 2002).

Atualmente, existem ontologias que possibilitam a compreensão dos modelos conceituais e funcionamento dos sistemas. Entretanto, ainda existem dificuldades que consistem em interpretação, manipulação e conceitualização para analisar os modelos de bancos de dados extraídos por engenharia reversa com o auxílio da ontologia, bem como a definição dos modelos conceituais obtidos, para tornar clara a compreensão dos mesmos.

Existe uma diversidade sistemas de informação, com seus bancos de dados, que não possuem uma construção adequada em relação à documentação e codificação e que necessitam ser explorados pela necessidade de se compreender os processos utilizados, que consistem na utilização do sistema que manipula os bancos de dados e de que maneira os usuários entendem as informações que são inseridas e, principalmente, os dados armazenados. Estes sistemas de informação geralmente possuem grandes volumes de dados e apresentam dificuldades de extração de conhecimento

Estes modelos de dados conceituais podem ser trabalhados para uma melhor compreensão através do auxílio da ontologia no mapeamento e validação das metapropriedades, possibilitando uma melhor definição dos processos e acarretando uma melhor utilização dos dados e, posteriormente, uma aplicação de mineração de dados.

1.1 Objetivos

O presente trabalho tem como objetivo geral propor uma metodologia de construção de um modelo conceitual de dados, com o propósito de possibilitar mineração de dados a partir de uma engenharia reversa do modelo físico, mapeamento e validação das metapropriedades do modelo conceitual, com o auxílio da ontologia.

Perguntas a serem respondidas:

1) É possível uma extensão das metodologias existentes de ontologia para mapear e validar o modelo conceitual?

2) O modelo gerado, através da metodologia proposta, é passível de ser aplicado a diversos algoritmos de mineração de dados?

1.2 Principais Contribuições do Trabalho

A primeira contribuição é a utilização das ontologias para direcionar a compreensão, mapear e validar a construção de um modelo conceitual.

A segunda contribuição é gerar uma metodologia que possa construir um modelo de dados compreensível com auxilio de ontologia, a partir de um modelo físico de dados, que apresenta extrema dificuldade e esforço hoje em sua definição.

A terceira contribuição é a utilização do modelo gerado para mineração de dados

1.3 Organização do Trabalho

Este trabalho está dividido da seguinte forma: no Capitulo 2 são apresentados ontologias e editores de ontologia; no Capítulo 3, conceitos de mineração de dados; o Capítulo 4 apresenta a proposta de construção da metodologia de um modelo de dados conceitual com o auxílio da ontologia; no Capitulo 5 implementa-se o estudo de caso; no Capítulo 6, encontram-se as conclusões e trabalhos futuros.

2. ONTOLOGIAS

A palavra Ontologia, definida originalmente na Filosofia como a ciência que estuda “os tipos de coisas que existem no mundo”, tem sido utilizada com diferentes sentidos na área de Inteligência Artificial e Engenharia do Conhecimento, o que tem provocado certa descaracterização do termo.

MOREIRA (2003) faz uma discussão detalhada sobre os significados do termo Ontologia na Filosofia e na Ciência da Computação. No sentido filosófico, Ontologia pode ser definida, em linhas gerais, como a teoria do ser, que se ocupa de responder às perguntas “quem existe?” e “que é consistir?”. O ramo da ontologia que se ocupa em responder à primeira pergunta é a Metafísica e o que responde à segunda questão é chamada de Teoria do Objeto, Teoria da Objetividade ou Teoria da Consistência dos Objetos em Geral (MORENTE, 1964). A Ontologia, dentro da filosofia de Aristóteles, teve também influência na Ciência da Computação e está associada a dez categorias básicas que são utilizadas para classificar tudo o que existe no mundo e que revelam a sua visão ontológica deste.

Ainda segundo MOREIRA (2003), o termo Ontologia começou a ser utilizado na Ciência da Computação, mais precisamente na subárea de Inteligência Artificial (IA), em projetos para organização de grandes bases de conhecimento, no início dos anos 90, quando houve um impulso para criação de bases de conhecimento compartilháveis e reutilizáveis, a fim de reduzir custos.

JASPER e USCHOLD (1999), citados por MOREIRA (2003), identificaram quatro categorias principais de aplicações para Ontologias na Ciência da Computação:

Autoria Neutra: a informação é criada em um único idioma, sendo convertida em formatos diferentes para uso em diferentes sistemas.

Ontologia como Especificação: ontologia com informações específicas sobre um domínio é criada e utilizada como base para especificação e desenvolvimento de software.

Acesso Comum à Informação: ontologia torna a informação, expressa em vocabulário pouco conhecido ou em formato inacessível, inteligível, provendo um entendimento compartilhado dos termos.

Busca Baseada em Ontologia: uma ontologia é usada para procurar recursos

desejados em um repositório de informações (como documentos ou páginas *Web*).

Apesar de o termo Ontologia estar presente em diversas áreas da Ciência da Computação, ainda não existe um consenso sobre o seu significado.

GUARINO e GIARETTA (1995), com o objetivo de fornecer um esclarecimento a respeito do significado do termo "Ontologia", apresentam algumas possíveis interpretações para o mesmo, sendo visto tanto como entidade semântica conceitual quanto como um objeto sintático específico. No entanto, os autores concentram-se numa análise minuciosa da definição de Ontologia fornecida por GRUBER (1993), como sendo "uma especificação explícita de uma conceitualização". Esta definição tem sido a mais utilizada pela comunidade da área de Inteligência Artificial e pode ser classificada como uma interpretação sintática para Ontologia (GUARINO e GIARETTA, 1995), uma vez que trata Ontologia como um produto concreto sintático e manipulável.

MOREIRA (2003) mostra um exemplo para ilustrar as noções de conceitualização e especificação. Uma conceitualização da visão do mundo formada pelo domínio acadêmico poderia conter os conceitos *aluno, professor, disciplina, nota, curso* etc. Uma especificação explícita desta conceitualização poderia ser constituída de sentenças descritas em lógica de primeira ordem, como na Tabela 1.

Tabela 1 – Especificação explícita de uma conceitualização. Fonte: MOREIRA (2003)

Sentença em Lógica	**Linguagem Natural**
$\forall x\ disciplina(x) \Rightarrow \exists y(professor(y) \wedge ministra(x,y))$	Para toda disciplina existe um professor que a ministra.
$\forall x\ disciplina(x) \Rightarrow \exists y(curso(y) \wedge parte\text{-}de(x,y))$	Toda disciplina é parte de um curso.
$\forall x\ aluno(x) \Rightarrow \exists y(curso(y) \wedge cursa(x,y))$	Todo aluno cursa um curso.
$\forall x\ aluno(x) \Rightarrow Corpo\text{-}acadêmico(x)$	Todo aluno é parte de um corpo acadêmico.
$\forall x\ professor(x) \Rightarrow Corpo\text{-}acadêmico(x)$	Todo professor é parte de um corpo acadêmico.
$\forall x\ aluno(x) \wedge Inteligente(x)$	Existem alunos que são inteligentes.

De forma semelhante, GUARINO e GIARETTA (1995) propõem atualização do termo "conceitualização" para denotar uma estrutura semântica que reflete um sistema conceitual, dentro de um aspecto intencional, ou seja, uma relação descrevendo todos os possíveis estados particulares de um mundo; e o termo "teoria ontológica" para denotar a teoria lógica, como vista na Tabela 1, utilizada para expressar conhecimento ontológico, no nível sintático. Assim, o termo "ontologia",

compatível com a definição de GRUBER (1993), corresponde a uma teoria ontológica, desde que esteja claro que o significado da palavra “explícito”, em tal definição, seja um objeto concreto, em nível simbólico.

GUARINO e GIARETTA (1995) afirmam que um conjunto de restrições formais, expresso por meio da teoria ontológica e utilizado a fim de se limitar as extensões admissíveis de uma conceitualização, pode caracterizá-la apenas parcialmente, uma vez que tais restrições podem representar diferentes modelos do mundo (conceitualizações). O conjunto de tais modelos, representados pela teoria ontológica, é que os autores definem como *compromisso ontológico*.

A fim de proporcionar um melhor esclarecimento do que vem a ser *compromisso ontológico*, na definição de GUARINO e GIARETTA (1995), será apresentado um exemplo, semelhante ao mostrado no referido artigo.

Seja T1 uma teoria lógica, conforme descrita abaixo:

$\forall x\ cachorro(x) \supset \forall\ animal(x)$

$\forall x\ gato\ (x) \supset \forall\ animal(x)$

$cachorro(Rex)$

$gato(Tob)$

A teoria ontológica T2, subjacente a T1, é obtida isolando-se o conteúdo ontológico de T1, tomando somente as sentenças que traduzem conhecimento intencional, ou seja, que sejam verdadeiras em todos os estados do mundo, e que podem ser vistas como uma tradução de parte do significado de *cachorro*, *gato* e *animal*:

$\forall x\ cachorro(x) \supset \forall\ animal(x)$

$\forall x\ gato\ (x) \supset \forall\ animal(x)$

Uma teoria ontológica, como T2, caracteriza apenas superficialmente o conteúdo ontológico de T1. A fim de proporcionar uma melhor caracterização do seu conteúdo, deve-se observar a conceitualização pretendida, subjacente a T1 e T2, que modela os aspectos ontologicamente relevantes da linguagem utilizada pela teoria inicial T1. Tal conceitualização pode ser (parcialmente) caracterizada, utilizando-se operadores modais, como o operador $\Box$, cujo significado é “em todos os possíveis mundos”, conforme mostrado em T3, abaixo:

$\Box\ (\forall x\ cachorro(x) \supset \forall\ animal(x))$

□ *(∀x gato (x) ⊃ ∀ animal(x))*

□ *(∀x cachorro(x) ⊃ □ cachorro(x))*

□ *(∀x gato(x) ⊃ □ gato(x))*

□ *(∀x animal(x) ⊃ □ animal(x))*

¬□ *(∀x pelo(x) ⊃ □ pelo(x))*

Esta teoria expressa algumas restrições gerais sobre o significado dos predicados: *cachorro, gato, animal* e *pelo*. T3 é dita ser, então, a especificação do *compromisso ontológico* de T1.

Baseados no que foi acima exposto, GUARINO e GIARETTA (1995) colocam que o termo "ontologia", segundo a definição de GRUBER (1993), pode ser interpretado como "uma teoria lógica que define explícita e parcialmente uma conceitualização".

VALENTE (1995), citado por MOREIRA (2003), define *comprometimento ontológico* como sendo as escolhas que levaram a selecionar um determinado conjunto de conceitos dentro de um determinado domínio, em detrimento de outro. Com base nesta definição de *comprometimento ontológico*, e de forma semelhante à interpretação de ontologia proposta por GUARINO e GIARETTA (1995), MOREIRA (2003) afirma que o registro explícito e formal dos *comprometimentos ontológicos* é o que tem sido normalmente denominado de Ontologia, na área de Inteligência Artificial.

MOREIRA (2003) salienta que, conforme discutido em GUARINO (1995) e POLI (2001), deve-se atentar para o tipo de conhecimento embutido em uma ontologia. Muitas vezes, o conhecimento no nível *ontológico* é confundido com conhecimentos dos níveis *lógico* e *epistemológico*. No nível lógico, o conhecimento é codificado por meio de primitivas básicas, como predicados e funções, como mostra a primeira coluna da Tabela 1. No nível epistemológico, o conhecimento é estruturado, relacionando-se seus conceitos através de primitivas de estruturação, como mostrado na segunda coluna da Tabela 1. Já no nível ontológico, os compromissos ontológicos associados aos conceitos relativos ao conhecimento são especificados explicitamente (GUARINO, 1995), sendo fornecida uma descrição da natureza destes conceitos. Um exemplo de conhecimento do nível ontológico, ainda no domínio acadêmico, seria "Todo aluno é um *objeto material*" e "Inteligente é uma *qualidade*" (MOREIRA, 2003).

2.1 Ontologias Formais

Ontologia Formal pode ser definida como o registro dos compromissos ontológicos feitos através de uma linguagem formal, com uma semântica bem definida (MOREIRA, 2003).

COCHIARELLA (1991) definiu Ontologia Formal como sendo "o desenvolvimento sistemático, axiomático e formal de todas as formas e modos de ser".

GUARINO (1995) afirma que, de acordo com a definição de COCCHIARELLA (1991), Ontologia Formal está relacionada à descrição rigorosa das características estruturais dos objetos e que, na prática, pode ser vista como a teoria das distinções entre as entidades do mundo (objetos físicos, eventos etc) e as categorias nível-meta utilizadas para modelar o mundo (conceito, propriedade, qualidade, estado etc).

Ontologias formais podem abranger desde termos gerais, que formam o fundamento para a representação de conhecimento em todos os domínios, denominadas "Ontologias de Nível Topo", até termos que são restritos a domínios de conhecimentos específicos, denominadas "Ontologias de Domínio".

2.2 Ontologias de Nível Topo

Ontologias de nível topo são utilizadas para classificar os elementos de uma ontologia de domínio, aperfeiçoando, assim, o entendimento de suas estruturas e seus relacionamentos. Seu objetivo principal é fornecer um "framework" para organização dos objetos de qualquer domínio.

GUARINO e WELTY (2000a, 2000b) mostram como uma ontologia de nível topo pode ajudar na utilização correta de relações hierárquicas entre elementos de um domínio, o que depende de uma compreensão correta da natureza das propriedades do domínio correspondentes aos nós taxonômicos da ontologia. A fim de facilitar tal compreensão, os autores apresentam algumas metapropriedades derivadas de noções filosóficas, que, combinadas de maneira sistemática, formam uma ontologia de nível topo.

DEGEN et. al. (2001) apresentam a ontologia de nível topo subjacente à linguagem de modelagem *GOL* (*General Ontology Language*). A linguagem *GOL* pode ser vista como uma extensão de linguagens de modelagem ontológica padrão, tais como

KIF (GENESERETH e FIKES, 1992) e similares, que são limitadas aos princípios de construção da teoria de conjuntos. *GOL* mantém teoria de conjuntos como parte de sua ontologia nível topo, aceitando relação de elementos de teoria de conjuntos, além de introduzir várias outras relações ontologicamente básicas e tipos.

O modelo ontológico apresentado por BUNGE (1977, 1979), com o objetivo de fornecer uma ontologia exata e científica, também consiste em uma ontologia de nível topo, uma vez que trabalha com elementos abstratos de alto nível que objetivam representar todos os fenômenos do mundo real. Tal modelo é baseado em tradições ontológicas provenientes da Filosofia e em pesquisas atuais, sendo elaborado por meio de conceitos matemáticos. A categoria mais geral neste modelo são "*coisas"* (*things*), que se referem a coisas concretas ou entidades reais do mundo. "*Coisas*" possuem *propriedades*. Uma *propriedade* pode ser "*intrínseca"*, o que significa que se aplica a apenas uma "*coisa*", ou "*mútua"*, significando que depende de duas ou mais "*coisas*". Um exemplo seria o "peso" de uma pessoa, que consiste em uma propriedade intrínseca, uma vez que depende apenas da existência da pessoa. Já a propriedade "ser um estudante" seria uma propriedade mútua, uma vez que depende da existência de uma pessoa e também de uma escola. Uma *classe* é definida por um conjunto de "*coisas*" que possuem uma propriedade em comum e um *tipo* é definido por um conjunto de *propriedades*. "*Coisas"* similares podem ser representadas pelo mesmo modelo, sendo que um modelo específico de uma "*coisa"* consiste num *esquema funcional*. "*Coisas"* podem interagir e podem também ser associadas, para formar uma outra "*coisa"*.

RUSSELL e NORVIG (1995) também apresentam uma ontologia de nível topo cujas categorias mais gerais são *Objetos Abstratos* e *Eventos*. *Objetos abstratos* são divididos em *Conjuntos*, *Números*, e *Objetos Representacionais*. *Eventos* são classificados em *Intervalos*, *Lugares*, *Objetos Físicos* e *Processos*. A classe de *Categorias* é uma subclasse da classe de C*onjuntos.* Esta ontologia inclui a relação de instanciação, identificada como "*membership*", e a relação "*parte-de*". Um *Evento* consiste num acontecimento com extensão temporal e espacial. Um *Intervalo* é um evento que inclui como subeventos todos os eventos ocorridos em um dado período de tempo.

2.3 Ontologias de Domínio

Ontologias de domínio modelam informações relativas a domínios de conhecimento específicos, através da incorporação de termos pertencentes a estes em uma ontologia de nível topo.

Nesse sentido, ontologias de domínio podem ser vistas como uma descrição detalhada da natureza de elementos pertencentes a domínios específicos, podendo ser utilizadas como fonte de conhecimento do domínio e, consequentemente, como ferramentas de suporte para a tarefa de modelagem conceitual de sistemas.

2.4 Metodologias

Esta análise tem como objetivo relacionar as metodologias utilizadas na construção de ontologias, identificando a forma com a qual cada uma dá suporte à construção do conhecimento. Outro objetivo é identificar nessas metodologias a construção do conhecimento, partindo-se de um conjunto estruturado de dados em um banco de dados relacional.

Ainda serão analisados aspectos relevantes em cada uma das metodologias, com a finalidade de caracterizá-las ou não como uma metodologia de fato. Para tanto, será considerada a afirmação de que uma metodologia deve incluir técnicas, métodos e princípios para ser considerada abrangente (USCHOLD e GRUNINGER, 1996).

2.4.1 Metodologia CyC

A metodologia Cyc foi iniciada em 1980 na *Microeletronics and Computer Technology*. Esta metodologia caracteriza-se por considerar o conhecimento consensual sobre os eventos cotidianos no mundo. Incluem-se neste consenso regras e heurísticas para dedução sobre objetos e os eventos (CYC PROJECT, 2005).

Muitas são as aplicações que se integram à base de conhecimento, sendo relevante a esta pesquisa o sistema de integração de bases de dados heterogêneos, na qual há o mapeamento dos vocabulários Cyc para os esquemas de bases de dados (FERNANDEZ, GOMEZ-PÉREZ e CORCHO, 2005).

O desenvolvimento da base de conhecimento CyC é realizado através de três processos, definidos em 1990: extração manual do conhecimento, extração auxiliada

por computador e extração gerenciada por computador (LENAT e GUHA, 1990). Durante a execução dos três processos, são desenvolvidas as atividades de criação de uma ontologia de alto nível com conceitos abstratos e a atividade de refinamento da ontologia abstrata, em busca de uma ontologia que represente o conhecimento desejado. O processo de extração manual do conhecimento considera essencial a leitura pelo homem de fontes como livros e artigos, destacando as informações que fazem tais fontes serem consideradas relevantes e identificando questões que possam ser respondidas ao ler o texto. Todo esse conhecimento deve ser inserido manualmente, através da codificação em linguagem CycL na base de conhecimento Cyc. O segundo e o terceiro processos são realizados com auxílio de meios computacionais, em que há a capacidade de obtenção de novos conhecimentos de senso comum.

É importante observar que o processo de construção do conhecimento apresentado pela metodologia Cyc não contempla o ciclo de vida exigido para a construção de conhecimento a partir do modelo físico dos bancos de dados relacionais. Não há definição dos processos necessários à obtenção do conhecimento, levando-se em consideração as restrições e as regras impostas pelos modelos matemáticos utilizados nos Sistemas de Gerenciamento de Banco de Dados Relacionais. A metodologia Cyc cita o mapeamento de conhecimento para esquemas de banco de dados, porém existe a necessidade de uma metodologia que defina os passos para obtenção do conhecimento a partir dos esquemas físicos do banco de dados.

2.4.2 Metodologia de Gruninger e Fox

A metodologia Gruninger e Fox teve como base a experiência obtida no projeto *Toronto Virtual Enterprise.* Foi desenvolvida em 1995, por Michael Gruninger e MarkS. Fox (GRUNINGER e FOX, 1995). A metodologia de Gruninger e Fox tem como objetivo criar, a partir do senso comum, uma ontologia que represente um modelo de conhecimento compartilhado no domínio empresarial, deduzindo deste domínio questões e respostas (GRUNINGER e FOX, 1995). As questões representam os problemas empresariais e o modelo a ser desenvolvido representa as respostas a estas questões. Por utilizar lógica de primeira ordem na especificação das ontologias, a

metodologia Gruninger e Fox é considerada formal, o que é vantajoso ao ser comparado com a capacidade de expressão da lógica clássica.

As ontologias desenvolvidas por este método visam criar modelos organizacionais que tenham a capacidade de fornecer uma terminologia compartilhada, na qual exista uma definição semântica de cada termo, implementando a semântica em um conjunto de axiomas que deduzem as respostas às questões do domínio da organização.

Esta metodologia se baseia na descrição de problemas dentro do escopo empresarial. A essa descrição dá-se o nome de cenário de motivação. Com base no cenário motivacional, é possível chegar a um conjunto de soluções. Os cenários também formam a base que define os requisitos para construção da ontologia. Com o uso da linguagem natural e com base no conjunto de cenários motivacionais, são desenvolvidas as questões de competências. As questões de competência são importantes na verificação da ontologia, pois é possível verificar se o conjunto de questões pode ser respondido pelo modelo proposto (GRUNINGER e FOX, 1995).

Com o uso das questões de competência formal e as respostas em linguagem natural formuladas, é possível extrair uma base de conhecimento. Tal base é inserida nas definições formais, nos axiomas e nos conceitos. Com isso, é possível obter uma linguagem que expressa as definições e restrições, propiciando uma terminologia que é utilizada no tratamento das questões de competência informal (GRUNINGER e FOX, 1995).

No modelo, as entidades são representadas por objetos organizados em taxonomias, que especificam suas propriedades e relações. Observa-se que, no desenvolvimento da ontologia, é necessária a identificação dos objetos que pertencem ao domínio e suas propriedades. Os objetos são representados por constantes e variáveis. As propriedades representam os conceitos dos objetos.

A metodologia Gruninger e Fox propõe dois passos a serem seguidos durante o processo de formalização. Num primeiro momento, deve ser realizada a formalização das questões de competência, quando são criadas regras em linguagem formal que definem semanticamente as sentenças envolvidas nas questões. Num segundo momento, devem ser especificados os axiomas formais, quando as definições semânticas dos termos e suas restrições são especificadas nas ontologias (GRUNINGER e FOX, 1995). Vale ressaltar que a especificação dos axiomas ainda não faz parte do processo de implantação, mas sim do processo de especificação da ontologia.

A verificação da ontologia envolve um conjunto de testes relativos ao domínio da ontologia. Os testes são definidos tendo como base a especificação dos requisitos, da modelagem e da formalização. A validação da ontologia deve observar a capacidade que o modelo e as implementações construídas com base nesse modelo possuem em responder às questões de competência.

A metodologia proposta por Gruninger e Fox é específica para construção de ontologias baseadas na organização de uma empresa, como, por exemplo, a ontologia que descreve as atividades de uma organização (GRUNINGER e FOX, 1995). A metodologia Gruninger e Fox não atende, desta forma, à necessidade de construção de ontologias tendo como base o conjunto de informações contido no modelo físico presente nos bancos de dados relacionais.

2.4.3 Método KACTUS

A motivação do projeto KACTUS está no compartilhamento e reuso das bases de conhecimento em diversos sistemas. O conhecimento é organizado utilizando a ontologia de domínio de forma independentemente da aplicação a ser construída (BERNARAS, LARESGOITI e CORREA, 1996).

Durante o processo de construção da aplicação, ocorre a adaptação de ontologias existentes, refinando suas características, de modo que atenda ao objetivo da aplicação. Caso não seja possível realizar este processo, uma nova ontologia deve ser construída no nível de domínio e refinada para a aplicação em questão.

Dentro desse processo devem ser realizadas três tarefas: especificação da aplicação; projeto preliminar; refinamento e estruturação da ontologia. A etapa de especificação da aplicação deve ter como resultado uma lista de termos e tarefas que será utilizada para criação dos componentes da ontologia. O projeto preliminar tem como objetivo a obtenção de visões globais baseadas no modelo, podendo ser feito através da reutilização de ontologias existentes ou da criação de novas ontologias, quando as existentes não satisfazem as necessidades do domínio. A etapa de refinamento e estruturação tem como objetivo a obtenção de uma ontologia definitiva, utilizando os princípios de organização hierárquica e modularização construída (BERNARAS, LARESGOITI e CORREA, 1996).

Esta metodologia possui um nível de detalhamento muito pequeno, mostrando apenas uma ideia daquilo que deve ser feito em cada fase. É uma metodologia voltada para a integração de ontologias e não descreve especificamente os passos necessários para construção do conhecimento durante a construção de uma ontologia. As fases deste processo não são documentadas devidamente e não há uma etapa de avaliação das ontologias resultantes. Essa metodologia não atende aos requisitos para construção de ontologias a partir do modelo físico de banco de dados relacional. Não há, em momento algum, descrições de tais etapas. A metodologia KACTUS está fortemente ligada à reutilização de ontologias existentes.

2.4.4 Metodologia Proposta por Uschold e King

A proposta inicial do método foi feita por Mike Uschold e Martin King em 1995(USCHOLD e KING, 1995), sendo complementada por Mike Uschold e Michael Gruninger (USCHOLD e GRUNINGER, 1996). A metodologia proposta é voltada à construção de ontologias no domínio empresarial.

A construção da ontologia é realizada dentro de um processo que envolve estágios de identificação do escopo e do propósito da ontologia, construção da ontologia com captura, codificação e integração e estágios finais de avaliação e documentação.

A identificação do propósito da ontologia visa principalmente esclarecer o uso pretendido com a ontologia. Isto é alcançado com a identificação da necessidade da construção da ontologia, levando em consideração o compartilhamento e o reuso de parte de um conhecimento. O formalismo que define o grau de formalidade sobre os termos utilizados, o propósito que conduz à intenção de uso e o assunto da ontologia que mostra o domínio do conhecimento onde a ontologia será aplicada caracterizam os seus três aspectos fundamentais (USCHOLD, 1996).

A fase de captura da ontologia visa como resultado a obtenção de um modelo de domínio prévio antes da construção e implementação da ontologia. Nesta fase, é concebida uma organização de termos numa taxonomia, que é alcançada através da definição exata dos termos, conceitos e as relações existentes entre eles. Uschold e Kingdefinem definem como critérios do método de captura da ontologia a identificação

nos métodos sobre o alcance dos conceitos, o nível de detalhamento e a facilidade no aprendizado (USCHOLD e KING, 1995).

A aquisição do conhecimento ocorre com a identificação das relações entre os conceitos principais do domínio e suas definições precisas, produzindo um vocabulário consensual, no qual são identificados os termos, conceitos e relações. O conhecimento é representado tendo como base essas definições (USCHOLD e GRUNINGER, 1996). É recomendada a produção, em linguagem natural, de um documento que especifique termos e frases relevantes no domínio. Tal documento ajuda na comunicação e na elaboração do escopo da ontologia.

São propostas três abordagens na tarefa de identificação dos conceitos da ontologia. Na primeira abordagem, são identificados os termos mais abstratos, os quais são especializados até se encontrar a definição dos termos mais específicos. Na segunda abordagem, é feito o contrário - primeiro são identificados os termos especialistas e aplicam-se os conceitos de generalização até encontrar os conceitos abstratos. Na terceira abordagem, que é recomendada pelos autores por resultar em modelos estáveis, é identificado inicialmente um termo básico, o qual é especializado ou generalizado quando for necessário (USCHOLD e KING, 1996).

A implementação da ontologia tem como objetivo a representação dos conceitos capturados nas fases anteriores através da codificação em uma linguagem capaz de representar os aspectos ontológicos da ontologia.

Esta metodologia possui um nível de detalhamento muito pequeno, apresentando apenas suas fases e um conjunto de ideias daquilo que pode ser feito em cada uma delas, não descrevendo especificamente os passos necessários à construção da ontologia. Essa metodologia não atende aos requisitos para construção de ontologias a partir do modelo físico de banco de dados relacional, pois não há descrição de tais etapas. A metodologia proposta por Uschold e King está fortemente ligada à construção de ontologias no domínio empresarial, berço de sua origem (USCHOLD e KING, 1995).

2.4.5 Metodologia Sensus

Sensus é uma metodologia para desenvolvimento de ontologia que é derivada da ontologia SENSUS (SWARTOUT, 1996). A ontologia SENSUS possui um grande número de termos e conceitos organizados em uma hierarquia. A metodologia Sensus é utilizada para especializar tais termos em domínios particulares, os quais não são atendidos pela ontologia SENSUS, A metodologia é, na verdade, um conjunto de processos que conduz o desenvolvimento da nova ontologia, ligando termos do domínio aos termos amplos de alto nível da ontologia SENSUS (SWARTOUT, 1996).

Num primeiro momento, é necessário identificar os termos relevantes na construção da ontologia de domínio dentro da ontologia SENSUS, como também identificar os termos relevantes e que fazem parte apenas do domínio. Após, é necessário ligar os termos do domínio que está sendo construído com os termos da ontologia SENSUS, incluindo também todos os conceitos que estiverem entre os termos. O processo é realizado de forma recorrente até que a ontologia de domínio esteja completa.

Esta metodologia possui um nível de detalhamento muito pequeno, apresentando apenas suas fases e um conjunto de ideias daquilo que pode ser feito em cada uma delas, não descrevendo especificamente os passos necessários para a construção da ontologia. Esta é uma metodologia de desenvolvimento de ontologias em domínios específicos através da técnica de especialização de conceitos. Esta metodologia não atende aos requisitos para construção de ontologias a partir do modelo físico de banco de dados relacional, pois não há em momento alguma descrição de tais etapas. A metodologia proposta está fortemente ligada à especialização da ontologia SENSUS em domínios de conhecimento específicos (SWARTOUT, 1996).

2.4.6 Metodologia 101

A metodologia 101 foi concebida através da experiência de Deborah L. McGuinness e Natalya F. Noy no desenvolvimento de uma ontologia de vinhos (NOY e McGUINNESS, 2001). Tem como base o paradigma de orientação a objetos e trata os objetos do mundo real como conceitos, dando ênfase à semântica.

A metodologia proposta é composta de quatro atividades: definição de classe, definição da taxonomia, definição dos *slots* e adição de valores aos *slots.* Tais atividades

estão inseridas dentro de um ciclo de vida composto dos seguintes passos: definição do escopo, verificação do reuso, seleção dos termos, definição de classes, atribuição das propriedades, definição das restrições e criação das instâncias (NOY e McGUINNESS, 2001).

Determinar o escopo da ontologia deve ser a primeira etapa do processo de construção. Nessa etapa, é importante descobrir o domínio, os usuários, a finalidade e também aquilo que a ontologia a deve responder. Com o escopo definido através das questões anteriores, é importante verificar a possibilidade de reutilização de ontologias existentes.

A seleção dos termos que irão compor a ontologia é realizada através de uma série de questionamentos com os quais se buscam a identificação dos termos, suas propriedades e a semântica utilizada em sua definição.

Com a seleção e definição dos termos concluídas, é necessário realizar a definição das classes dentro de uma hierarquia. As classes são definidas a partir do agrupamento dos termos anteriores, levando em consideração a similaridades entre eles. A hierarquia é a classificação das classes dentro de uma taxonomia. O processo de organização e definição da hierarquia das classes segue princípios semelhantes aos utilizados nas definições de classes na linguagem UML, acrescentando a definição de conceitos e semântica.

A descrição da estrutura interna da classe, também referenciado como descrição dos conceitos, faz parte do processo de criação da ontologia, mais especificamente da etapa de definição e atribuição de propriedades. As propriedades são atribuídas a uma classe a partir da lista de termos, que foi gerada na etapa de seleção de termos. Os *slots* dentro da metodologia são caracterizados por descrever os atributos de instâncias e suas relações. A definição dos atributos (propriedades) de uma classe também envolve a etapa de definição de restrições do domínio de valores possíveis que podem ser utilizados em um *slot.*

Dentro do processo de construção da ontologia apresentado na metodologia 101, as autoras definem a última etapa do processo como a criação das instâncias. Nesta etapa, são atribuídos valores dos *slots* das instâncias individuais geradas a partir de uma classe.

O processo de construção do conhecimento apresentado pela metodologia 101 não contempla o ciclo de vida para a construção de conhecimento a partir do modelo físico dos bancos de dados relacionais. A metodologia 101 descreve o processo de

construção de classes, que tem origem no paradigma da orientação a objetos, adicionando a este a representação dos objetos em forma de conceitos (NOY e McGUINNESS, 2001). Não há definição dos processos necessários à obtenção do conhecimento, levando em consideração as restrições e as regras impostas pelos modelos matemáticos utilizados nos modelos físicos de bancos de dados, não podendo, portanto, ser aplicada para este fim.

2.4.7 Metodologia Methontology

A Methontology é uma metodologia para construção de ontologia desenvolvida entre 1996 e 1997, no laboratório de inteligência artificial da Universidade Politécnica de Madri. Participaram do desenvolvimento os pesquisadores Mariano Fernández, Asuncion Gomez-Pérez e Natalia Juristo (FERNÁNDEZ, GOMEZ-PÉREZ e JURISTO, 1997). Baseia-se no desenvolvimento de sistemas adotando um ciclo de vida evolutivo. A metodologia é composta por um conjunto de estágios de desenvolvimento, com técnicas de planejamento, gerenciamento e suporte.

Na etapa de planejamento, são definidos as tarefas a ser desenvolvidas, a forma de utilização dos recursos, prazos, formas de controle e garantia da qualidade da ontologia que está sendo construída.

O estágio de especificação dos requisitos contempla tanto as atividades de especificação, quando são definidos os motivos pelos quais está sendo construída a ontologia, o escopo, os possíveis cenários e casos de usos. Também são definidos os usuários da ontologia (FERNÁNDEZ, GOMEZ-PÉREZ e JURISTO, 1997).

A tarefa de aquisição do conhecimento é executada durante todas as etapas do processo, sendo que suas atividades são executadas mais frequentemente durante o estágio de especificação. A aquisição do conhecimento é realizada através da consulta e análise de livros e entrevistas com especialistas do domínio da ontologia.

O estágio de conceitualização ou modelagem conceitual é responsável pela descrição dos problemas e sua solução. Durante o processo de modelagem conceitual, serão desenvolvidos todos os conceitos, instâncias, relações, axiomas, atributos etc. Estes artefatos compõem os termos do domínio. Neste estágio também serão desenvolvidas as representações intermediárias, resultando em um dicionário de dados,

árvores de classificação de conceitos, tabelas de atributos de instâncias, tabelas de atributos de classe, tabelas de constantes, tabelas de fórmulas e um conjunto de classificação de atributos em árvores (GOMEZ-PEREZ, FERNANDEZ, VICENTE, 1996).

O estágio de formalização não é obrigatório, mas pode ser realizado utilizando-se lógica descritiva. A fase de implantação pode ser realizada através de uma linguagem formal, como o Prolog (FERNANDEZ, GOMEZ_PEREZ e CORCHO, 2003).

O estágio de avaliação pode ser executado durante os estágios iniciais, fazendo com que erros não se propaguem. Neste estágio, serão verificados se aquilo que foi planejado foi executado corretamente e se o resultado atende aos requisitos iniciais.

Todos os estágios de execução da construção da ontologia seguindo este método devem ser documentados. A manutenção ocorre sob demanda, sendo executada quando necessário.

Ao se analisar esta metodologia, é possível observar que o processo de construção é bem detalhado, sendo que há um ciclo de vida definido, contando também com atividade de gerenciamento, desenvolvimento e suporte. O método leva à construção de uma ontologia de qualidade. A metodologia Methontology descreve o processo de criação de ontologia semelhante aos processos de desenvolvimento de software baseados na prototipação (FERNÁNDEZ, GOMEZ-PÉREZ e JURISTO, 1997). Porém, o processo de construção do conhecimento apresentado pela metodologia Methontology não contempla o ciclo de vida exigido para a construção de conhecimento a partir do modelo físico dos bancos de dados relacionais. Não há definição dos processos necessários à obtenção do conhecimento, levando em consideração as restrições e as regras impostas pelos modelos matemáticos utilizados nos sistemas de gerenciamento de banco de dados relacionais, não podendo ser aplicada a este fim.

2.4.8 Metodologias Baseadas em Engenharia Reversa

De acordo com Chikofsky e Cross (1990), pode-se definir engenharia reversa como uma coleção de teorias, metodologias e técnicas capazes de suportar a extração e abstração de informações de um software existente, produzindo documentos consistentes, quer seja a partir do código-fonte, ou através da adição de conhecimento e experiência que podem ser automaticamente reconstruídos a partir do código. De acordo

com Meersman, Spyns e Jarrar (2002) e Gómez-Pérez e Manzano-Macho (2003), a engenharia reversa pode ser vista como um conjunto de métodos e técnicas usado para construir uma ontologia através de processos automáticos ou semiautomáticos de aquisição de conhecimento via textos, dicionários, bases de conhecimento, dados semiestruturados e esquemas relacionais já existentes.

Algumas metodologias que se utilizam da engenharia reversa para construção de ontologias estão presentes na literatura. Entre elas, estudou-se a metodologia de Sugumaran e Storey (2002), a metodologia de Bueno (2005), a metodologia de Saias e Quaresma (2003) e a metodologia de Dogan e Islamaj (2002).

2.4.9 Metodologia de Sugumaran (2002)

No trabalho de Sugumaran e Storey (2002), a metodologia é baseada em identificar e definir os termos, as propriedades, os relacionamentos e os relacionamentos necessários para modelar um domínio da aplicação. Esta metodologia é composta por 4 passos:

1. identificar a base de termos do modelo entidade-relacionamento (MER), que permitirá identificar os termos mais frequentes na base de dados e usar o especialista para identificação dos sinônimos e termos relativos;

2. identificar os tipos de relacionamentos entre os conceitos;

3. criar as regras e associações entre os termos;

4. ajudar o especialista a fazer uma identificação de alto nível para finalizar a captura do domínio de conhecimento e formar definitivamente a ontologia. Os indivíduos são inseridos após a construção propriamente dita da ontologia, consultando a base de dados.

2.4.10 Metodologia de Bueno (2005)

Para Bueno (2005), a metodologia de construção de ontologias de engenharia reversa consiste em 7 passos.

1. relacionar todo domínio, ou seja, catalogar todas as fontes de informações digitais que servirão como base de dados do sistema;

2. aplicar o Extrator de Frequência de palavras sobre a base de dados

relacionada;

3. realizar a comparação entre os resultados dos extratores com as necessidades dos especialistas;

4. construir junto com o especialista um vocabulário controlado representativo do domínio a ser investigado;

5. utilizar o vocabulário definido no passo anterior para aplicar o extrator semântico na base de dados;

6. avaliar o resultado com base na frequência das expressões indicativas encontradas e definir uma lista de palavras;

7. construir a ontologia para utilização no sistema com base no vocabulário controlado.

2.4.11 Metodologia de Saias (2003)

Saias e Quaresma (2003) desenvolveram uma metodologia contendo 5 passos. É uma ferramenta que permite explorar um grupo de documentos, analisando e organizando as palavras de acordo com sua freqüência:

1. realizar a definição inicial da ontologia a partir de uma base de dados;
2. identificar os conceitos e as propriedades referentes ao MER;
3. estabelecer as relações entre os conceitos, consultando um especialista;
4. criar a ontologia utilizando os 3 passos anteriores;
5. validar a ontologia. Neste passo, surge um *merge* entre a ontologia criada recentemente e uma ontologia obtida de outra base de dados com o mesmo domínio.

2.4.12 Metodologia de Dogan (2002)

A metodologia de Dogan e Islamaj (2002) foi criada a partir de um modelo entidade-relacionamento (MER) e sem interação do usuário. A engenharia reversa, simples e inteiramente automática, é composta por apenas 3 passos:

1. transformar as relações em conceitos;
2. transformar os atributos nas relações em atributos nos conceitos;

3. transformar os registros na base de dados relacional para indivíduos da ontologia.

2.4.13 Metodologia Ontoclean

Guarino e Welty propõem uma metodologia para construção de ontologias de domínios, denominada OntoClean, que tem como objetivo ser um guia na construção de ontologias (GUARINO e WELTY, 2002).

Os princípios empregados na metodologia estão relacionados principalmente ao objetivo de melhorar o relacionamento taxonômico dentro da ontologia. Para isso, é necessário identificar de forma clara os conceitos empregados na construção da modelagem do domínio, verificando assim se tais relacionamentos estão sendo utilizados de forma correta. (GUARINO e WELTY, 2002).

As metapropriedades apresentadas por Guarino e Welty representam o comportamento das propriedades, tendo como objetivo promover o entendimento das propriedades do domínio. O benefício da utilização da metodologia OntoClean é a capacidade de visualização das propriedades mais importantes. Com isso, é possível identificar quais relacionamentos taxonômicos estão sendo feitos de forma errada.

A base de desenvolvimento da OntoClean é composta por noções de identidade, essência, unidade e dependência, que sofreram um processo de combinação, gerando as metapropriedades utilizadas pela metodologia OntoClean. Estas metapropriedades visam facilitar a compreensão correta da natureza das propriedades pertencentes a um domínio.

A metapropriedade *Rigidez* é originada a partir da noção de *Essência*, que está relacionada ao fato de uma propriedade se aplicar a um elemento do domínio (também chamado de instância da referida propriedade), enquanto tal elemento existir.

Uma propriedade pode ser classificada, com relação à metapropriedade *Rigidez*, da seguinte forma:

- Propriedade Rígida (denotada pelo símbolo $+R$) → propriedade que é essencial para todas as suas instâncias, ou seja, qualquer elemento do domínio que instancia tal propriedade permanecerá instanciando-a durante toda a sua existência;

- Propriedade Não-Rígida (denotada pelo símbolo -*R*) → propriedade que não é essencial para alguma de suas instâncias, ou seja, poderá existir alguma instância de um elemento do domínio que não permanecerá instanciando a propriedade durante toda a sua existência;
- Propriedade Antirrígida (denotada pelo símbolo ~*R*) → propriedade que não é essencial para todas as suas instâncias, ou seja, todas as instâncias de um elemento do domínio necessariamente não permanecerão instanciando a propriedade durante toda sua existência;
- Propriedade Semirrígida (denotada pelo símbolo ¬*R*) → propriedade não-rígida (-R), porém que não seja antirrígida (~R).

Pode-se deduzir que uma propriedade antirrígida é também uma propriedade não-rígida, porém o primeiro conceito é mais forte que o último, uma vez que restringe todas as instâncias de uma propriedade, enquanto o último restringe no mínimo uma instância.

A metapropriedade *Rigidez* não é herdada ao longo da hierarquia de propriedades, o que permite que uma propriedade rígida subjugue uma propriedade não-rígida.

Para melhor esclarecimento, considere, por exemplo, as propriedades *Funcionário* e *Pessoa*. Obviamente, todas as instâncias de *Funcionário* são também instâncias de *Pessoa*. Porém, uma instância da primeira propriedade poderá deixar de ser um funcionário, uma vez que poderá ficar desempregado ou se tornar um empresário ou trabalhador autônomo, mas jamais deixará de ser uma instância de pessoa. Isto nos faz concluir que a propriedade *Funcionário* é não-rígida (-R), sendo mais precisamente antirrígida (~R), pois todas as instâncias de *Funcionário* não serão necessariamente para sempre instâncias de *Funcionário*; e a propriedade *Pessoa* é rígida (+R), pois todas suas instâncias permanecerão como tal por toda a sua existência.

A metapropriedade *Dependência Externa* é originada a partir da noção filosófica de Dependência. Uma propriedade φ é externamente dependente, denotada pelo símbolo *+D (-D*, caso contrário), de uma propriedade ψ, se, para todas as suas instâncias *x*, necessariamente deve existir alguma instância de ψ que não seja parte nem constituinte de *x*.

As relações de parte e constituinte são discutidas com maiores detalhes em

GUARINO e WELTY (2000b).

Por exemplo, a propriedade *Cliente* é externamente dependente da propriedade *Empresa*, pois uma pessoa somente poderá ser um cliente se houver uma empresa da qual ela adquire produtos ou serviços prestados. Já a propriedade *Pessoa* não é externamente dependente de um coração ou corpo, pois uma instância de *Pessoa* possui um coração como parte e é constituída por um corpo. Para entendimento das metapropriedades baseadas na noção de Identidade, primeiro deve-se definir o conceito de Condição de Identidade (CI) para uma propriedade φ, que consiste numa relação ρ, satisfazendo a seguinte fórmula:

$$\varphi(x) \square \varphi(y) \rightarrow (\rho(x,y) \leftrightarrow x=y)$$

Dessa forma, uma propriedade "executa uma CI", denotada pelo símbolo *+I* (*-I*, caso contrário), se existirem, em tempos distintos, instâncias que: 1) se satisfazem a mesma CI, então elas são iguais (CI suficiente); ou, 2) se são iguais, então satisfazem a mesma CI (CI necessária).

Além disso, uma propriedade φ "fornece uma CI", denotada pelo símbolo *+O* (*-O*, caso contrário), apenas se ela for rígida (+R), executar uma CI (+I), e esta não for executada por nenhuma propriedade que a subjugue. Isto significa que, se φ herda CIs diferentes (mas compatíveis) de múltiplas propriedades, ela ainda permanece fornecendo um CI.

CIs são herdadas ao longo da hierarquia de propriedades. Assim, uma propriedade não-rígida pode executar uma CI se e somente se esta for herdada de uma propriedade rígida que a subjuga.

Por exemplo, a propriedade *Cliente*, que é não-rígida, pode apenas executar suas CIs (+I), herdando-as de propriedades rígidas que a subjugam, como a propriedade *Pessoa* (+O). Isto se dá devido ao fato de uma mesma pessoa poder ser cliente em diferentes tempos de diferentes empresas. Assim, uma CI fornecida por cliente, como, por exemplo, ter o mesmo código, pode ser apenas local; enquanto que uma CI fornecida pela propriedade *Pessoa*, como ter o mesmo número de Carteira de Identidade ou as mesmas impressões digitais, terá validade global. Já a propriedade *Verde* não executa CI (-I), pois não existe CI necessária nem suficiente que identificará ou reidentificará coisas verdes, pelo simples fato de serem verdes.

Uma propriedade que executa uma CI (+I) é chamada de "ordenável"

(STRAWSON, 1959).

Para reconhecer que uma propriedade é “ordenável”, não é necessário saber qual CI ela executa: a distinção entre “ordenável” e não “ordenável” é normalmente suficiente para estabelecer ordem em taxonomias (GUARINO e WELTY, 2000a, 2000b).

GUARINO e WELTY (2000c, 2001) fornecem discussões mais detalhadas sobre o conceito de identidade.

Para entendimento das metapropriedades baseadas na noção de Unidade, primeiro deve-se definir o que significa, para uma certa propriedade, possuir uma Condição de Unidade (UC), isto é, constituir-se um *todo*:

Seja ω uma relação de equivalência. Em um dado tempo *t*, um objeto *x* é um *todo contingente* sob ω se cada parte de x estiver ligada por ω a todas as suas outras partes e a nada mais.

Uma noção mais forte de *todo* pode ser definida assumindo-se que uma UC deve sustentar um objeto por toda a sua existência: “Seja ωuma relação de equivalência. Um objeto *x* é um *todo intrínseco* sob ω se, a qualquer tempo em que *x* exista, ele é um *todo contingente* sob ω”.

Dessa forma, uma propriedade “*executa uma UC*”, denotada pelo símbolo *+U*, se e somente se exista uma relação de equivalência ω tal que todas as suas instâncias são todos intrínsecos sob ω.

GUARINO e WELTY (2000b, 2000d) e WELTY e GUARINO (2001) salientam que, como ocorre com a metapropriedade Rigidez, pode ser útil, em alguns casos, distinguir entre propriedades que não executam uma UC comum para todas as suas instâncias daquelas propriedades cujas instâncias não sejam *todos intrínsecos*. Um exemplo do primeiro caso seria a propriedade *Agente Legal*, de que todas as instâncias constituem-se *todos intrínsecos* (algumas pessoas, algumas empresas), porém não existe uma única relação ω para todas elas (desde que pessoas e empresas podem ter diferentes UCs). *Monte de Matéria* é normalmente um exemplo do segundo caso, desde que nenhuma de suas instâncias constitui-se em *todo intrínseco*.

Uma propriedade possui “*antiunidade*”, denotada pelo símbolo *~U*, se cada uma de suas instâncias não constitui *todos intrínsecos*.

Uma propriedade “não possui unidade”, denotada pelo símbolo *–U*, se ela não executa uma UC comum para todas as suas instâncias, ou se suas instâncias não se constituem em *todos intrínsecos* (~U).

GUARINO e WELTY (2000c) discutem mais detalhadamente os conceitos ligados à *Unidade.* Ainda nesse artigo, os autores fazem a seguinte definição: “Uma propriedade que executa uma CI (+I) e também executa uma UC (+U) é chamada de *contável*”.

Através das combinações de rigidez, unidade, identidade e dependência, é possível se chegar a um conjunto de classes de propriedades, apresentadas na Tabela 2: categorias, tipos, quase tipos, papéis formais, papéis materiais, sortais com fases, atribuições e mixins. (GUARINO e WELTY, 2002).

As categorias, dentro de uma taxonomia, não podem ser subjugadas por propriedades ordenáveis, pois estão geralmente no topo da taxonomia (GUARINO e WELTY, 2002).

Categorias são propriedades rígidas (+R), porém não executam uma CI (-I). Dessa forma, elas não podem ser subjugadas por nenhuma propriedade ordenável, o que as leva a estarem presentes normalmente nos níveis mais altos em uma taxonomia, sendo utilizadas para fins classificatórios.

De acordo com as restrições vistas anteriormente, *Categorias* podem ser subjugadas por outras *Categoria*s e *Atribuições*, e podem subjugar quaisquer tipos de propriedades. Segundo GUARINO e WELTY (2000a), *Categorias* tendem a formar uma árvore, sendo recomendado que as categorias de nível mais alto sejam disjuntas. Exemplos de *Categoria* podem ser: *Entidade*, *Entidade Concreta* e *Entidade Abstrata*.

Tipos são propriedades rígidas (+R) e que fornecem sua própria CI (+O). Como eles são os únicos que fornecem CI, de acordo com o Princípio da Individualização Ordenável, cada elemento do domínio modelado deve instanciar pelo menos um Tipo. Assim, estes representam as propriedades mais importantes de um domínio.

Em uma hierarquia, de acordo com as restrições vistas anteriormente, Tipos podem ser subjugados por Categorias, outros Tipos, Quase-Tipos e Atribuições. Eles podem subjugar quaisquer tipos de propriedades ordenáveis. GUARINO e WELTY (2000a) recomendam que Tipos sejam subjugados por, no mínimo, uma Categoria. Exemplos de Tipos são: *Pessoa, Gato* e *Água*.

Quase-Tipos são propriedade rígidas (+R) que apenas executam e não fornecem

identidade (+I -O). Quase-Tipos podem ser utilizados para agrupar entidades do domínio, baseados em combinações de propriedades úteis que não afetam identidade.

Em uma taxonomia, de acordo com as restrições vistas anteriormente, Quase-Tipos podem ser subjugados por Categorias, Atribuições e Mixins e devem ser subjugados por, pelo menos, um Tipo, a fim de herdar a CI que executa. Quase-Tipos podem subjugar quaisquer tipos de propriedades ordenáveis. Exemplos de Quase-Tipos são: *Animal Invertebrado* e *Herbívoro*.

Papéis Formais consistem em propriedades que expressam a função desempenhada por uma entidade do domínio em relacionamento específico entre duas ou mais entidades. Todos os papéis são antirrígidos (~R) e dependentes (+D). Papéis Formais, adicionalmente, não executam CI (-I) e representam os papéis mais genéricos, que aparecem nos níveis mais altos da hierarquia de papéis.

Em uma hierarquia, de acordo com as restrições vistas anteriormente, Papéis Formais podem ser subjugados apenas por outros Papéis Formais, Atribuições ou Categorias, e podem subjugar apenas outras propriedades não-rígidas e dependentes. GUARINO e WELTY (2000a) recomendam que Papéis Formais sejam utilizados apenas para organizar uma hierarquia de papéis, subjugando apenas Papéis Materiais. Exemplos de Papéis Formais são: *Paciente* e *Instrumento*, no sentido de ser um paciente ou instrumento de uma ação, desde que não exista nenhuma condição de identidade em comum para estes (pois podem ser objetos, pessoas ou outros animais).

Papéis Materiais consistem em propriedades antirrígidas (~R) e dependentes (+D), mas que executam CI (+I), herdada de algum Tipo que a subjuga. Papéis Materiais representam papéis que tipos particulares de entidades do domínio desempenham em um evento.

Em uma hierarquia, de acordo com as restrições vistas anteriormente, Papéis Materiais podem ser subjugados por quaisquer tipos de propriedades, devendo ser obrigatoriamente subjugados por um Tipo, a fim de herdar a CI que executam. Eles podem subjugar outros Papéis Materiais e Mixins dependentes. GUARINO e WELTY (2000a) recomendam que Papéis Materiais subjuguem apenas outros Papéis Materiais e que sejam subjugados apenas por papéis e outras propriedades rígidas. Um exemplo típico de Papel Material é *Estudante*, que é subjugado pelo Tipo *Pessoa* e corresponde à participação desta em um evento *Matrícula*. Sortais com Fases representam entidades

independentes com identidade, que podem mudar com o tempo (GUARINO e WELTY, 2002).

Sortais com Fases consistem em propriedades que executam CI (+I), são antirrígidas (~R) e independentes (-D). Eles não fornecem uma CI global (O), pois fornecem apenas uma CI local, que corresponde a uma certa fase temporal de suas instâncias. Entidades deste tipo possuem CIs que mudam no decorrer do tempo e em fases discretas.

Em uma hierarquia, de acordo com as restrições vistas anteriormente, Sortais com Fases podem ser subjugados por quaisquer propriedades independentes, devendo obrigatoriamente ser subjugados por um Tipo, a fim de herdar a CI que executam, e podem subjugar propriedades não rígidas e que executem CI. GUARINO e WELTY (2000a) recomendam que Sortais com Fases sejam subjugados apenas por propriedades rígidas e que subjuguem apenas outros Sortais com Fases e Papéis Materiais. Estes autores também recomendam que os Sortais com Fases, que correspondem a todas as fases pelas quais passa uma determinada entidade do domínio, devem ser subjugados por um Tipo ou Quase-Tipo, que subjuga apenas esses. Outra restrição é que um Sortal com Fases nunca deve aparecer sozinho, pois sempre deve existir pelo menos um outro Sortal com Fase que corresponda a uma outra fase pela qual ele passa. Exemplos típicos de Sortais com Fases são *Lagarta* e *Borboleta,* que correspondem a fases pelas quais passa um determinado inseto durante a sua existência.

Atribuições consistem em propriedades que não fornecem e nem executam CI (-O, -I) e são antirrígidas (~R) e não dependentes (-D) ou semirrígidas (¬R). Elas representam os atributos ou qualidades das entidades de um domínio, como cor, forma, tamanho etc.

Em uma hierarquia, de acordo com as restrições vistas anteriormente, Atribuições podem subjugar quaisquer tipos de propriedades e podem ser subjugadas por quaisquer propriedades não ordenáveis. GUARINO e WELTY (2000a) recomendam que Atribuições subjuguem apenas Mixins e outras Atribuições, e sejam subjugadas apenas por categorias. Porém, no presente contexto, não faz sentido Atribuição fazer parte de uma hierarquia de propriedades, uma vez que representa qualidades pertencentes a elementos do domínio modelado. Exemplos de Atribuições seriam *Vermelho, Grande* e *Macho*.

Mixins consistem em propriedades que apenas executam CI (-O, +I) e são

semirrígidas (¬R). Estas propriedades representam várias combinações de propriedades rígidas e não-rígidas.

Em uma hierarquia, de acordo com as restrições vistas anteriormente, Mixins podem ser subjugados por quaisquer tipos de propriedades, devendo obrigatoriamente ser subjugados por, no mínimo, uma propriedade ordenável, e podem subjugar qualquer tipo de propriedade também ordenável. Mixins são pouco restringidos pelas metapropriedades vistas anteriormente, o que desencoraja, segundo GUARINO e WELTY (2000a), seu uso em uma ontologia, uma vez que podem gerar mais confusão na ordem na mesma. Exemplos de Mixins são *Gato-ou-Arma*, subjugando o Tipo *Gato* e o Papel *Arma.*

Tabela 2 Propriedades formadas a partir da combinação das metapropriedades
Adaptada de GUARINO e WELTY (2000).

Meta-Propriedades				Tipos de Propriedade	
+O	+I	+R	+D	*Tipo*	ORDENÁVEL
			-D		
-O	+I	+R	+D	*Quase-Tipo*	
			-D		
-O	+I	~R	+D	*Papel Material*	
-O	+I	~R	-D	*Ordenável com Fase*	
-O	+I	¬R	+D	*Mixin*	
			-D		
-O	-I	+R	+D	*Categoria*	NÃO ORDENÁVEL
			-D		
-O	-I	~R	+D	*Papel Formal*	
-O	-I	~R	-D	*Atribuição*	
		¬R	+D		
			-D		

Ao analisar esta metodologia, é possível observar que o objetivo principal da metodologia OntoClean está no fato de que ela proporciona mecanismos de desenvolvimento e validação dos relacionamentos entre classes de uma ontologia e das classes em si também. A ideia apresentada pelas noções de identidade, unidade, essência e dependência possuem uma relação em que é possível desdobramentos com o conceito de banco de dados. A metodologia OntoClean descreve o processo de criação de ontologia, baseado em metapropriedade, que definem as características dos relacionamentos taxonômicos (GUARINO e WELTY, 2002). Porém, o processo de construção do conhecimento apresentado pela metodologia OntoClean não contempla o ciclo de vida exigido para a construção de conhecimento a partir do modelo físico dos

bancos de dados relacionais. Entretanto, existe definição de características que possam ser aplicadas ao modelo de banco de dados relacionais, mas, considerando-se as restrições e as regras impostas pelos modelos matemáticos, não há definição de processo definido que possa ser aplicado para esse fim.

2.4.14 Metodologia SABiO

O método SABiO (Systematic Approach for Building Ontologies) é um artefato que pode ser utilizado para desenvolver uma ontologia de qualidade de software. Este método possui um processo sistemático baseado na abordagem da engenharia do conhecimento exteriorizado através de uma notação gráfica e sobre um ciclo de vida de desenvolvimento (FALBO, 1998).

A Metodologia SABiO possui um conjunto de etapas formais sequenciais que visa a construção da ontologia através de um fluxo principal guiado através da identificação do propósito e especificação de requisitos, seguido da captura e formalização da ontologia, tendo como produto final a ontologia formal. Em paralelo às etapas anteriores, ocorrem as etapas de avaliação e documentação e a integração com ontologias existentes.

Na primeira fase, ocorre a delimitação do escopo da ontologia, identificando aquilo que a ontologia pode responder e quais serão seus usuários. Estas atividades são realiazadas durante a etapa de identificação do propósito e especificação dos requisitos.

Durante a segunda fase, que é um processo interativo, são identificados os conceitos e relações, a taxonomia, os axiomas e o dicionário de dados. Nesta fase, ocorrem as definições das competências da ontologia.

Durante o desenvolvimento, pode haver a necessidade de realizar a integração com ontologias existente. Esta tarefa é realizada durante todo o processo de construção da ontologia, sendo assim uma atividade realizada em paralelo, visando sempre o aproveitamento daquilo que já foi conceitualizado.

A tarefa de avaliação visa identificar se aquilo que foi descrito na especificação foi atingido. Esta atividade também é desenvolvida em paralelo durante o processo de construção da ontologia, de forma iterativa.

A documentação tem como objetivo registrar todo o processo de desenvolvimento da ontologia. Nesta documentação, devem estar inclusos obrigatoriamente os objetivos

e propósitos, o cenário de motivação, a ontologia formal e todos os elementos produzidos e os utilizados durante a construção da ontologia.

Como pode ser observado, este processo de construção não atende aos requisitos de construção de ontologia a partir do modelo físico de uma base de dados. Embora o processo seja bem definido, não existe uma descrição sobre as etapas e procedimentos necessários para construção da ontologia que satisfaça ao que esta pesquisa propõe.

2.5 Editores de ontologia

2.5.1 APECKS (Adaptive Presentation Environment for Collaborative Knowledge Structuring)

O APECKS é um servidor de ontologias que permite a construção de ontologias de forma colaborativa. Através da colaboração dos indivíduos, é possível adaptar ontologias pessoais na construção do conhecimento (TENNISON e SHADBOLT, 1998). Uma ontologia pessoal representa o entendimento do domínio pelo indivíduo, ao passo que uma ontologia colaborativa é a representação do entendimento do domínio pelo grupo. O APECKS utiliza os recursos de rede, sendo ele um cliente e um servidor de ontologia, voltado ao apoio do especialista do domínio do conhecimento, mas que não seja necessariamente um especialista em criação de ontologia. O APECKS utiliza um sistema de desenvolvimento racional, em que cada alteração registra consigo os motivos e a lógica utilizada (TENNISON e SHADBOLT, 1998).

2.5.1.1 Características

O APECKS é um servidor de ontologias que permite trabalho cooperativo através da criação de ontologias pessoais pelos usuários. Estas ontologias podem ser comparadas com outras e é possível a discussão sobre as diferenças e similaridades entre elas (TENNISON e SHADBOLT, 1998).

A principal característica do APECKS está no apoio aos usuários através da comparação, enaltecendo o que há de consenso, correspondência e conflitos entre as ontologias comparadas. A ênfase do APECKS não é a ontologia gerada, mas todo o processo de construção da ontologia, com as discussões relacionadas à construção de uma ontologia consensual (TENNISON e SHADBOLT, 1998).

2.5.1.2 Arquitetura

O APECKS é baseado em uma arquitetura via internet, acessível e multiusuário, em ambiente virtual de texto, tendo com propósito propiciar facilidade de comunicação. A arquitetura é composta por duas partes, sendo que uma delas é um banco de dados orientado a objeto, que define cada objeto. O segundo é um servidor, que é encarregado por ler e interpretar o arquivo de banco de dados.

Em primeiro lugar, o banco de dados utilizado pelo APECKS é acessado via internet e utiliza o protocolo HTTP (hiper text transfer protocol), podendo agir como um servidor web. Em segundo lugar, o banco de dados gera HTML dinamicamente, com base na representação interna dos objetos. Em terceiro lugar, há uma série de objetos genéricos no banco de dados que formam a base de representação do conhecimento utilizado pelo APECKS (TENNISON e SHADBOLT, 1998).

2.5.1.3 Representação do conhecimento

A representação do conhecimento dentro do APECKS é feita pelo agrupamento de indivíduos dentro de uma classificação, que representa a hierarquia entre os indivíduos. Cada indivíduo possui *slots,* que são os locais onde podem ser definidos valores. Os indivíduos desse agrupamento podem realizar o processo de herança múltipla, no qual um *slot* pode herdar valores de outros indivíduos (TENNISON e SHADBOLT, 1998).

Algumas restrições se aplicam ao âmbito do APECKS. Como a representação do conhecimento destina-se apenas a uma prova de conceito, a hierarquia de classe é, em grande parte, determinada pela adesão das pessoas às classes, ao invés de serem

definidos diretamente pelo usuário. Os usuários não podem definir as facetas ou axiomas.

O conhecimento armazenado em um banco de dados APECKS é dividido em um número de *domínios*. Qualquer utilizador pode definir ontologias dentro de um domínio, representando diferentes aspectos do domínio ou diferentes tarefas que podem ser realizadas dentro dele. Cada uma dessas ontologias é conhecida como um papel (TENNISON e SHADBOLT, 1998).

2.5.1.4 Pontos fortes

A ontologia gerada é resultado da colaboração entre os indivíduos. O sistema também gera uma comparação entre os objetos definidos, evidenciando as diferenças e o consenso apontados pelos engenheiros das ontologias comparadas.

2.5.1.5 Pontos fracos

O APECKS é uma ferramenta capaz de gerar a ontologia através da colaboração dos usuários e este é o fator que motiva a construção do conhecimento. Pelo fato de utilizar banco de dados, a ferramenta não é capaz de gerar uma ontologia a partir de um modelo de dados existentes, mas cabe observar que este não é um dos propósitos da ferramenta.

2.5.1.6 Utilização

Dentro do APECKS, cada objeto possui um número de páginas *web* associadas, que traz informações essenciais sobre o objeto. Nestas páginas existem uma lista com as mudanças e um motor de busca. Também há tópicos de ajuda para todo tipo de objeto.

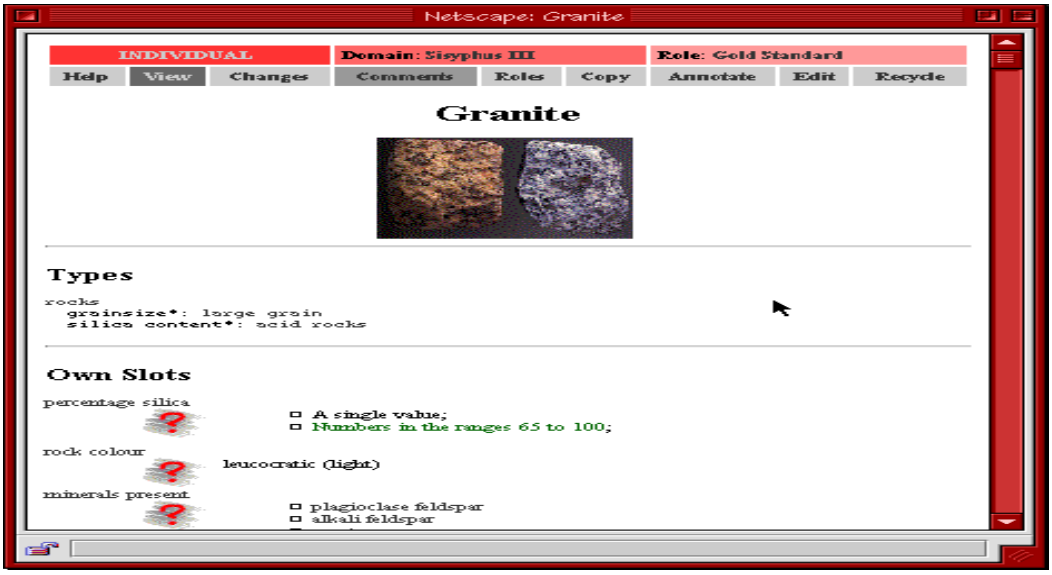

Figura 1: Exemplo de utilização do APECKS

A figura 1 representa a visualizaçao de um objeto dentro do APECKS. Cada objeto dentro do APECKS é representado por uma página, que traz informações sobre a sua existência.

2.5.1.7 Construção da ontologia

Primeiramente, o usuário constrói um papel sobre o qual ele vai representar sua visão sobre o domínio. O próximo passo é criar um objeto que armazenará informações sobre o domínio, criando indivíduos dentro do domínio. A aquisição de conhecimento na ontologia gerada pelo APECKS é resultado da inserção de dados pelo usuário ou da colaboração entre usuários.

2.5.1.8 Comparações entre ontologias

Uma das grandes vantagens do uso do APECKS está na capacidade de comparação entre as estruturas conceituais. A comparação entre as estruturas conceituais é feita através do consenso, conflito, correspondência e contraste. O uso do APECKS está baseado na colaboração para construção de ontologias. Sendo assim, ele possibilita a comunicação entre usuários durante o ciclo de vida da ontologia (TENNISON e SHADBOLT, 1998).

2.5.2 DOME - Domain Ontology Management Environment

O Dome (Domain Ontology Management Environment) é um ambiente para a manipulação de ontologias. Reconhece que o problema central na construção de serviços dinâmicos é a falta de métodos e ferramentas que suportem a integração de modelos de processo e sistemas de informação de múltiplas organizações em ambientes de processos compartilhados pelas empresas.

O principal objetivo é proporcionar os ambientes nos quais os desenvolvedores de ontologias possam utilizar ferramentas para desenvolver ontologias ou executar processos de engenharia reversa sobre fontes de dados e que usuários ou agentes de software possam utilizar as ontologias desenvolvidas para, dinamicamente, integrar múltiplos sistemas de informação (BRIAN e ZHAN, 2000).

A ontologia produzida pelo DOME consiste em termos denotados de conceitos, relacionamentos e restrições. Um conceito é caracterizado por atributos, que são condições necessárias, mas não suficientes. Esta definição é intencionalmente similar à definição de esquemas de banco de dados relacionais e classes de banco de dados orientados a objetos, uma vez que o DOME é centrado na construção de ontologias para fontes de dados estruturados (BRIAN e ZHAN, 2000).

2.5.2.1 Arquitetura

Baseando-se nos princípios da simplicidade, reusabilidade e perfeição, a suíte do DOME se compromete a resolver problemas das mais diversas áreas, incluindo ferramentas para edição, navegação e criação de ontologias. Desenvolvido por um grupo de trabalho para a gestão de ambientes de construção de ontologias, pode ser utilizado por qualquer profissional que busca uma ferramenta de gestão de ontologias.

Uma ontologia é vista como sendo um conjunto de conceitos orientados a um domínio, incluindo conceitos abstratos e restrições necessários para a consulta do conhecimento. É sobre esta definição que o DOME se baseia. Abaixo, a figura 2 representa a interação com as bases de dados.

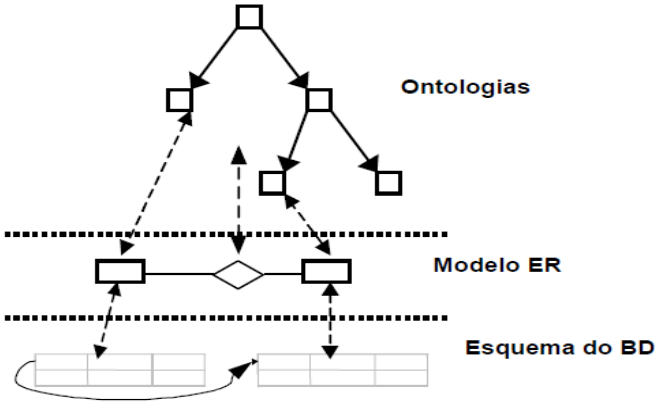

Figura 2: Arquitetura Dome

2.5.2.2 Características

O DOME distingue duas classes de termos de ontologias: termos primitivos e termos definidos. A semântica de termos primitivos não é especificada. Quando o mapeamento entre duas ontologias é necessário, tipos primitivos têm que ser mapeados manualmente. Nesse ambiente, uma ontologia forma uma hierarquia de especialização com os termos de mais baixo nível, tendo ligações estreitas com os modelos entidade-relacionamento entre ontologias, modelos ER e esquema de banco de dados (BRIAN e ZHAN, 2000).

- O DOME oferece ferramentas para extração de ontologias de sistemas legados, analisando os esquemas de banco de dados
- Editor e navegador sobre ontologias já existentes
- Ferramentas para o mapeamento entre ontologias e bancos de dados
- Checagem e validação de ontologias
- Possibilidade de juntar duas ontologias.

2.5.2.3 Pontos fortes

A extração de ontologias do DOME utiliza engenharia reversa a partir de fontes de dados e de seus programas de aplicação. Permite extrair entidades e relacionamentos a partir de esquemas de banco de dados, podendo ainda extrair relacionamentos semânticos e funções a partir de programas de aplicação. É importante salientar que muitos relacionamentos são determinados pelos programas de aplicação, de tal forma

que o uso correto dos dados de origem frequentemente não está documentado. A ontologia inicial precisa posteriormente ser refinada por projetistas de ontologias.

2.5.2.4 Pontos fracos

O DOME foi originalmente projetado para a manipulação de fonte de dados estruturados, mas cabe observar que uma ampliação proposta seria incluir as fontes de dados semiestruturados, tais como páginas *web* através de XML/DTD, o que daria uma abrangência maior para utilização.

2.5.2.5 Utilização

Funciona na forma de plugins para o Eclipse, além de contar com uma API genérica para acesso aos repositórios de ontologias. Abaixo, seguem algumas imagens do plug-in:

Na figura 3, é possível observar a integração com o editor. Observa-se também a definição das classes.

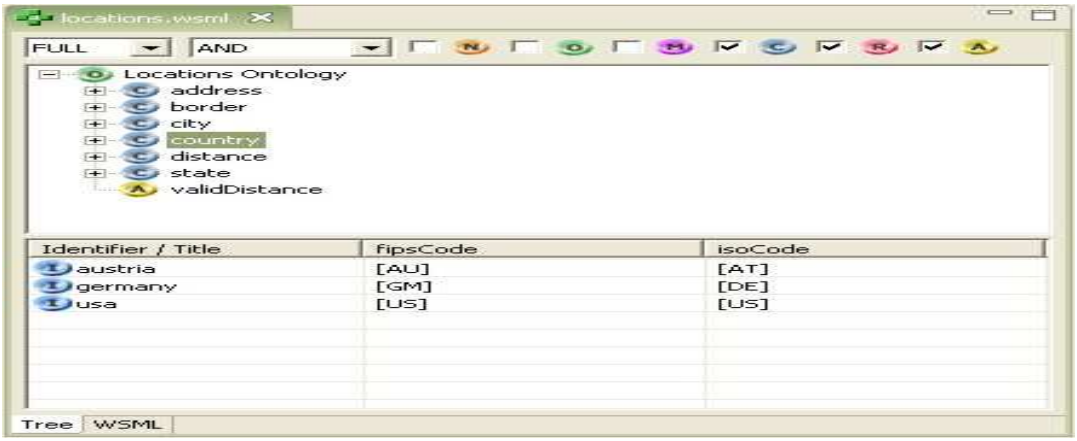

Figura 3: Integração com o editor

A figura 4 representa o processo de mapeamento dos dados através da árvore de mapeamentos.

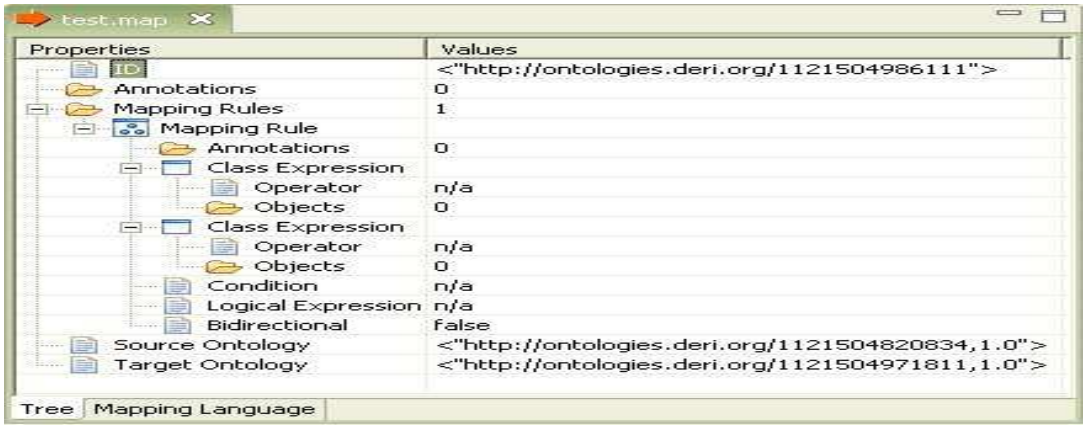

Figura 4: Mapeamento dos dados

2.5.3 JOE – Java Ontology Editor

Ontologias do ponto de vista do JOE são entidades-relacionamentos ou modelos de uma dada base de conhecimento. Neste momento do desenvolvimento, JOE não tenta validar a ontologia nem checa falhas no design – estes tipos de funcionalidades estarão nos planos futuros. Quando constrói ontologias, JOE suporta apenas estruturas ontológicas mínimas, como entidades, atributos e relações. Atualmente, o principal objetivo é prover uma interface gráfica para a representação de ontologias que podem ser usadas em um ambiente de código aberto (JOE, 2009).

2.5.3.1 Características

- Mostra todos os nodos disponíveis de uma dada ontologia, agrupados por entidades, atributos e relacionamentos.
- Possui painel para mostrar e editar as consultas de pesquisas
- Permite uma visualização da estrutura ontológica
- Permite a edição de ontologias de forma básica, como operações de adicionar conceitos e fazer relacionamentos.
- Opções de navegação, zoom e procura
- Permite a construção de consultas e o estabelecimento de restrições para uma determinada consulta.

2.5.3.2 Pontos fracos

O projeto aparenta ter sido descontinuado, sendo que sua última versão foi codificada sobre Java 1.1. Falta material bibliográfico e tutoriais estão indisponíveis no site do projeto.

2.5.3.3 Utilização

A figura 5 demonstra a execução do JOE em modo consulta. Nesse modo de execução, é possível executar buscas na ontologia.

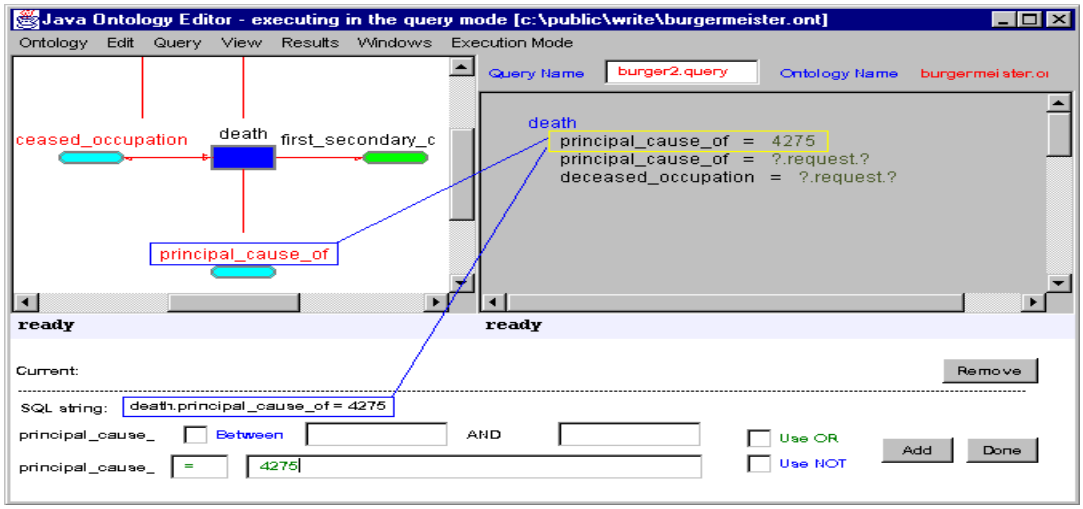

Figura 5: Execução em modo consulta do JOE

2.5.4 KMgen

O KMgen é um ambiente de edição de ontologias no qual é possível descrever um domínio de conhecimento específico através dos conceitos e relacionamentos existentes. Tal descrição deve ser realizada com a especificação dos termos e da semântica do domínio (KMGEN, 2010).

2.5.4.1 Características

Para criar a base de conhecimento, foi desenvolvida a linguagem de representação KM (Knowledge Machine), que é uma linguagem poderosa e com uma semântica clara, com lógica de primeira ordem. Essa linguagem possui mecanismos

sofisticados para o raciocínio, incluindo a seleção pela descrição, unificação, classificação e raciocínio sobre ações, usando um mecanismo de situações.

Os componentes são armazenados em um banco de dados relacional. A edição é feita com o auxílio do editor de textos Emacs. O KMgen é multiusuário e possui uma arquitetura de aplicação cliente-servidor. Um servidor pode estar no mesmo computador que o cliente ou em outro computador numa LAN ou na Internet (KMgen, 2010).

2.5.4.2 Pontos Fortes

KMgen possui diversas ferramentas para desenvolver a ontologia, sendo possível ter o banco de dados em diferentes plataformas. São elas:

- LispWorks, para o núcleo da aplicação (LispWorks é um software para o desenvolvimento em ANSI Common Lisp)
- Interface entre o Common Lisp e Java
- Java Standard Widget Toolkit (SWT), para a interface gráfica com o usuário
- PostgreSQL, o sistema de banco de dados relacional onde são armazenadas as informações sobre as ontologias que estão sendo desenvolvidas
- Emacs para edição
- CLSQL, a camada entre a Common Lisp e o sistema de banco de dados relacional.

2.5.4.3 Pontos Fracos

- Desenvolvimento somente em linguagem LISP
- Conexão somente com o banco de dados PostgreSQL
- Possui pouca documentação sobre o sistema.

2.5.4.4 Utilização

O painel esquerdo da janela KMgen exibe a hierarquia de frames selecionados.

A definição de um quadro é exibida em um navegador HTML (painel superior direito da janela principal KMgen). Ao clicar em um link (qualquer palavra colorida), traz um menu, como mostrado na figura 6.

A edição ocorre no Emacs, usando um modo específico KM (recuo, coloração e complementação de palavras), com exceção das manipulações das hierarquias, que são feitas usando-se comandos.

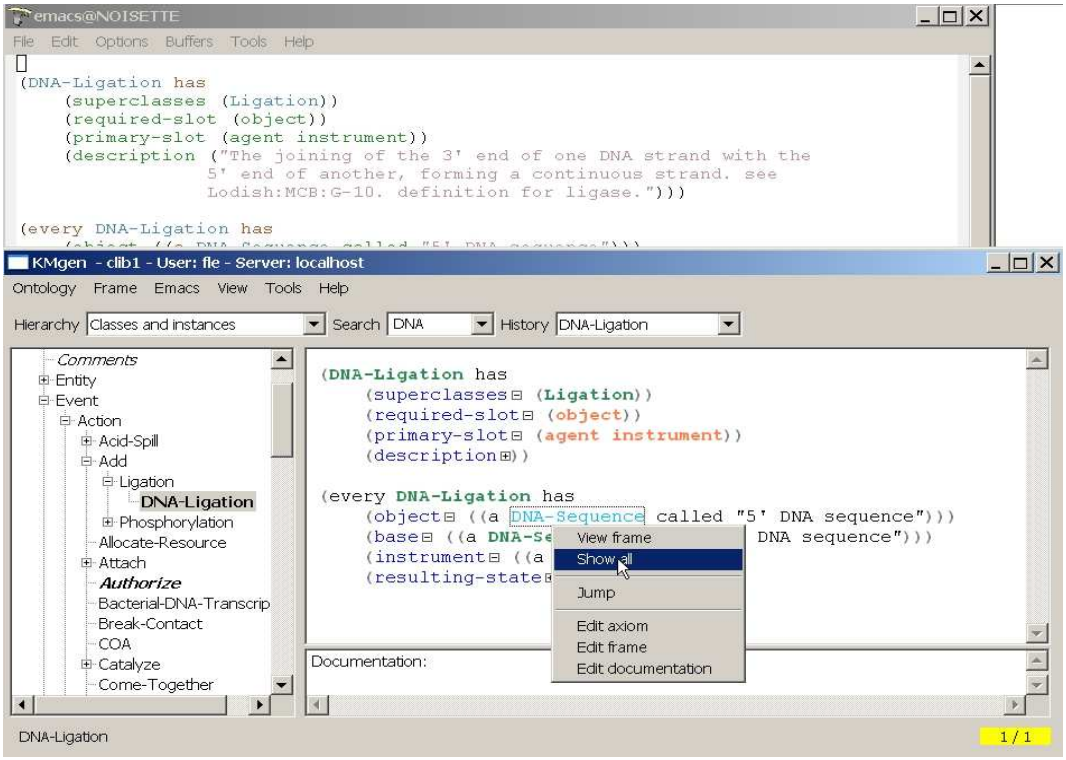

Figura 6: Utilização do KMgen

2.5.5 HOZO

HOZO possui um visualizador e editor de ontologias que permitem o controle de versionamento no desenvolvimento (KOZAKI ET AL, 2007). Nesse ambiente é assumido que as ontologias podem ser desenvolvidas numa atividade paralela, em que as ontologias podem ser dependentes umas das outras. O HOZO é um ambiente de desenvolvimento de ontologias distribuído e que permite a edição e a visualização da ontologia remotamente, mantendo uma dependência consistente entre as ontologias. Sendo assim, o HOZO oferece um ambiente de desenvolvimento que mapeia as dependências entre as ontologias (KOZAKI ET AL, 2007).

2.5.5.1 Arquitetura

A arquitetura do HOZO, conforme mostra a figura 7, é distribuída entre os diversos agentes que interagem na construção, camada de visualização e utilização da ontologia. Possui um servidor de ontologia, que é responsável por manter um repositório de ontologia. Possui uma camada de gerenciamento, que é responsável pela gestão das dependências entre as ontologias. A camada de gerenciamento está entre o servidor de ontologia e a camada de edição. A camada de edição é a responsável por dar suporte ao desenvolvimento e visualização da ontologia através do Onto Studio, um sistema de desenvolvimento de ontologia (KOZAKI ET AL, 2007).

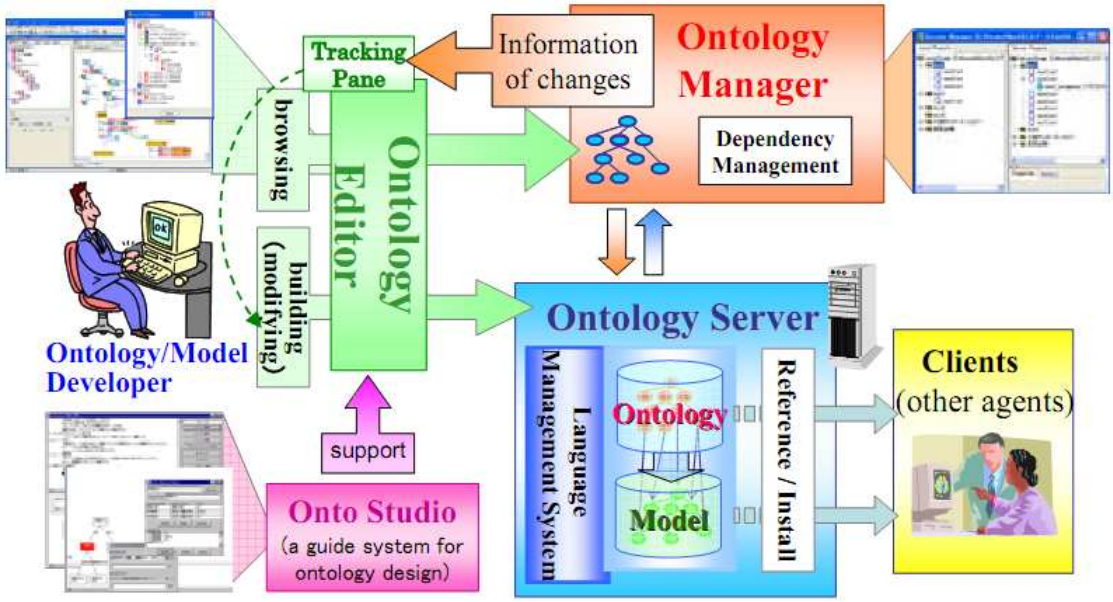

Figura 7: Arquitetura do HOZO

2.5.5.2 Características

Utiliza Onto-Estudio em sua base. Suporta os formatos abaixo para importação:

- CSV Text - o CVS é uma estrutura de dados semiestruturados disposta num arquivo de texto
- OWL é uma linguagem utilizada no processado da informação, usualmente em ambiente web (WOL, 2003).

Suporta os formatos abaixo para exportação

- XML-DTD - o XML provê a descrição sobre os dados e o DTD provê a estrutura do documento XML (W3C, XML Technology)

- DAML+OIL é o sucessor do DAM e do OIL, podendo ser utilizado para descrever um conjunto de fatores ao criar uma ontologia (DAM+OIL, 2001)

- RDF (Resource Descripton Framework) é uma linguagem utilizada para representar informações na web (W3C, 2004)

- OWL (web ontology language) é uma linguagem utilizada no processamento da informação, usualmente em ambiente web (WOL, 2004).

O HOZO possui um painel de administração de projeto que permite ao usuário administrar e visualizar a árvore do projeto e ontologias dentro do projeto. Possui painel de edição que permite ao usuário a troca entre várias ontologias com uma visualização em aba. Possui painel de navegação e exploração que permite uma visualização gráfica da Ontologia. Possui painel de definição que permite ao usuário editar o conteúdo de definição de uma ontologia.

2.5.5.3 Pontos fortes

- Funcionalidade de comunicação via internet serão lançados em breve na próxima versão
- Suporta formatos conhecidos
- Possui uma interface bem completa no quesito edição de ontologias
- É possível compartilhar a edição de ontologias, pois elas são armazenadas no servidor de ontologias e distribuídas para os desenvolvedores
- Para esse compartilhamento se tornar possível, existem as opções de segurança de edição, como *lock* e *unlock*, para prevenir múltipla sobrescrita.

2.5.5.4 Pontos fracos

- Não possui interação com o modelo físico de banco de dados relacional
- Limitado à construção de ontologias apenas. Não há nenhuma interação com modelos de dados em banco de dados.

2.5.6 OntoGloss

OntoGloss é uma ferramenta de anotação de ontologias que usa conceitos pré-definidos em uma linguagem de marcação. A diferença entre anotações regulares e anotações baseadas em ontologias é que a anotação é um texto simples, que é coletado baseado na estrutura fixada. A anotação é um conjunto de instâncias de classes e relacionamentos baseado na ontologia de domínio (MOSTOWFI, FOTOUHI, ARISTAR, 2005). Na anotação baseada em ontologia, o processo de anotação é o processo de relacionar e anotar texto para um conceito em uma ontologia (instanciando uma classe) ou para um tipo de dados ou, ainda, relatando-o para outro texto anotado (instanciando uma relação).

2.5.6.1 Arquitetura

O gerenciamento de ontologias e interface de exploração proveem uma interface genérica para diferentes representações de ontologias.

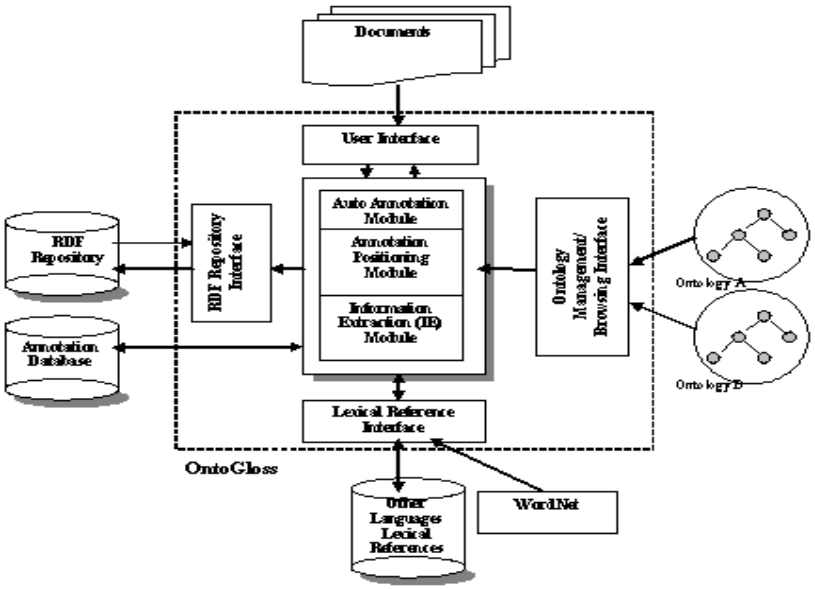

Figura 8: Arquitetura do OntoGloss

Conforme pode ser visto na figura 8, o OntoGloss possui uma interface genérica para representação da ontologia; um repositório RDF, que possui a função de fazer a

anotação sobre os dados; uma interface com o usuário, que é implantada através do Microsoft Access e do Internet Explorer, como pode ser visto na figura 9.

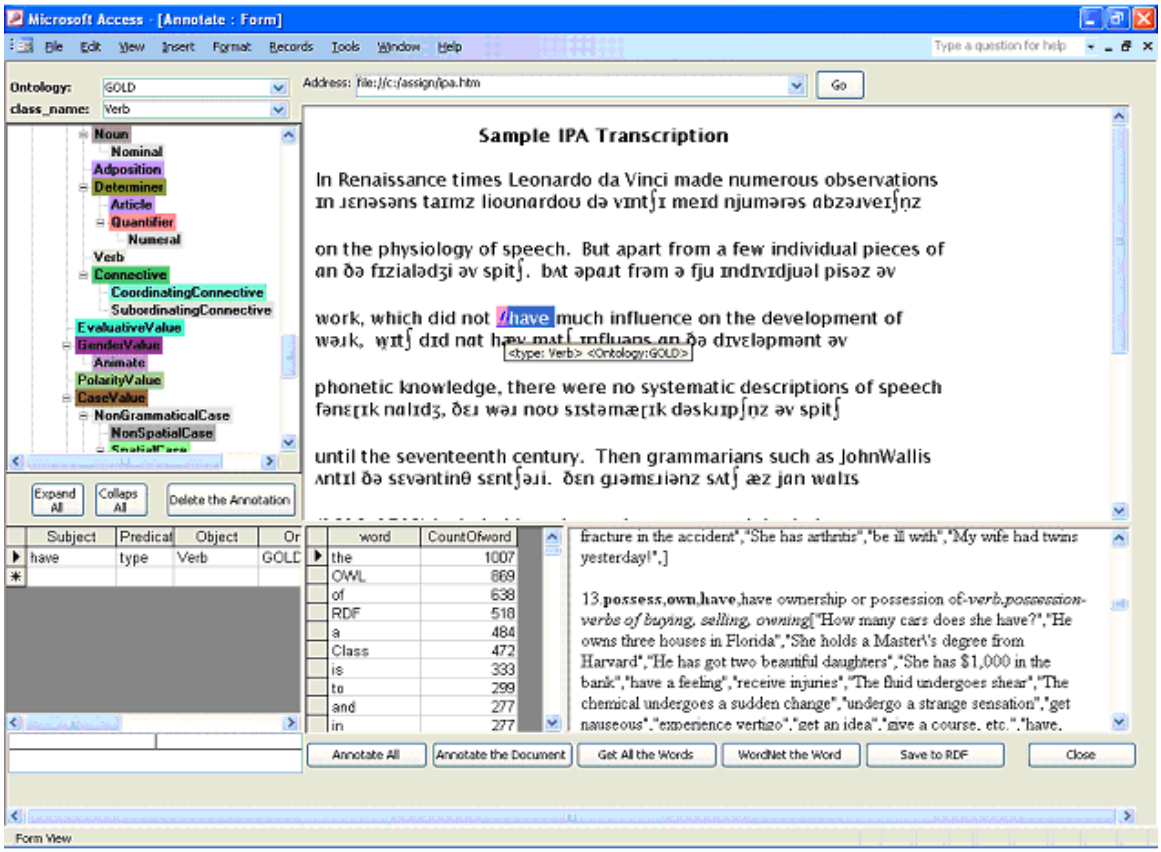

Figura 9: Exemplo de utlilização do OntoGloss

2.5.6.2 Características

- Usa diferentes ontologias para marcar documentos, parágrafos, sentenças e palavras
- Anota o documento com opções de arrastar e soltar
- Anota novos documentos baseados em documentos anotados anteriormente
- Habilidade para usar o sistema de referência
- Suporta servidores de anotações locais e remotos
- Suporta edição de OWL e RDF. OWL é uma linguagem utilizada no processamento da informação, usualmente em ambiente web (WOL, 2003). RDF é uma linguagem utilizada para representar informações na web (W3C, 2004)
- Exporta dados de anotações no formato RDF

- Dados que foram anotados podem ser salvos em banco de dados e carregados durante cada visita ao documento
- Permite anotar o documento inteiro com informações gerais, como o nome do anotador, data e outras informações.

2.5.6.3 Pontos fortes

- Possui bibliografia bem completa e arquivos de referência fáceis de entender
- Por usar anotações, pode-se obter o nível de granularidade da mais alta até a mais baixa, conforme necessário.

2.5.6.4 Pontos fracos

- Não interage diretamente com o modelo físico de dados, apenas com o *schema* do banco de dados
- Suporta poucos formatos.

2.5.7 OilED

OilED é um editor de ontologias desenvolvido por Sean Beachhofer na Universidade de Manchester. O propósito do editor é dar suporte à edição de ontologias com o uso da linguagem OIL (BECHHOFER ET AL, 2001). OilED não tem a intenção de ser um ambiente de desenvolvimento de ontologias completo. Não dá suporte ao desenvolvimento de ontologias em larga escala, à migração e à integração de ontologias, a versionamento e a várias outras atividades que são envolvidas com a construção de ontologias. Ao invés disso, oferece apenas as funcionalidades necessárias para permitir aos usuários construir ontologias e demonstrar como o FaCT (Fast Classification of Terminologies) pode ser usado para checar e enriquecer ontologias. FaCT, através da lógica de descrição, é utilizado na classificação dos conceitos em uma ontologia. (FACT, 2003).

2.5.7.1 Características

- Com o OielED, é possível construir ontologias
- Permite utilizar o Raciocinador FaCT para checar a consistência de ontologias e adicionar relações implícitas de subclasses (FACT, 2003)
- Permite a edição de arquivos RDF, DAML e OIL.

2.5.7.2 Utilização

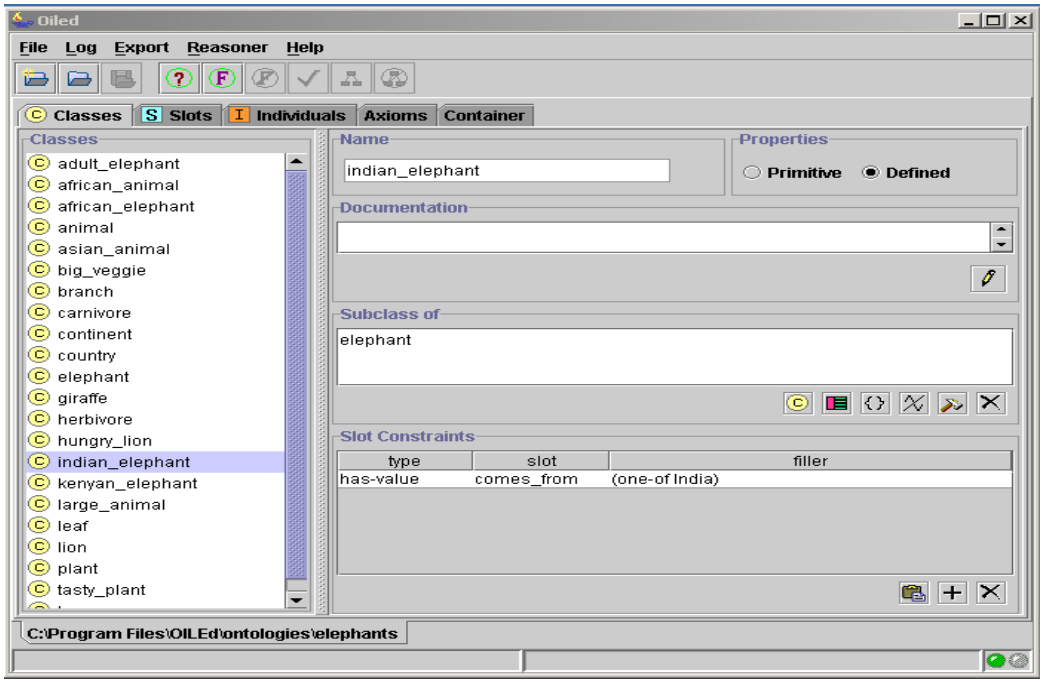

Figura 10: Tela de edição do OilED

A figura 10 mostra a tela de edição do Oiled. É possível observar as funcionalidades de edição de classes, slots e axiomas. Os axiomas são utilizados para definição de fatos sobre as classes (BECHHOFER ET AL, 2001).

2.5.7.3 Pontos fortes

- É uma ferramenta gratuita e de código-fonte aberto, podendo ser utilizada por qualquer pessoa
- Muito leve e rápido na hora de editar um arquivo que define uma ontologia.

2.5.7.4 Pontos fracos

- Não interage com servidores de ontologias
- Não interage com modelos físicos de banco de dados
- Possui poucas ferramentas para edição de ontologias.

2.5.8 ONTOEDITOR

A motivação para construção do OntoeEditor é a de prover uma ferramenta capaz de representar graficamente as ontologias, assim como armazená-las em um banco de dados relacional, num formato aprovado pela Web Semântica (VASCONCELOS, 2003).

2.5.8.1 Arquitetura

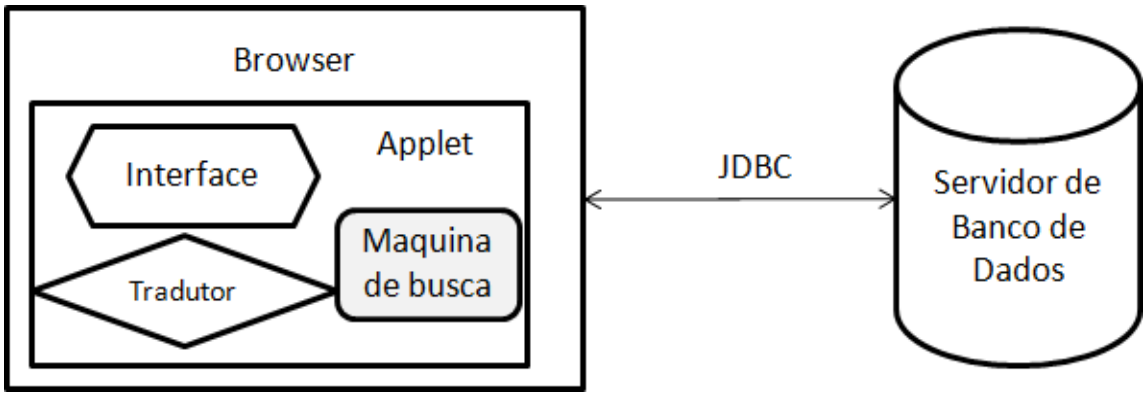

Figura 11: Arquitetura do OntoEditor

A aplicação é a camada que compreende a ferramenta. É através dela que o usuário final manipula suas ontologias na web. A camada de dados é o local onde são salvas as informações sobre as ontologias. Essa camada é implementada com a utilização de um banco de dados relacional, conforme figura 11 (VASCONCELOS, 2003).

2.5.8.2 Características

- É uma ferramenta utilizada para edição de ontologias via Internet, que implementa as visualizações arbóreas (*folder-tree),* hiperbólica (*Hyperbolic Tree)* e de grafos (*TouchGraph)*
- O ambiente de trabalho do OntoEditor está baseado na navegação web
- Possui a capacidade de importação de ontologias a partir de arquivos textos tabulados, que representam a estrutura da ontologia
- Interage com banco de dados relacional MySQL.

2.5.8.3 Pontos fortes

O usuário cria suas ontologias através da própria ferramenta e também pode armazenar suas ontologias no banco de dados e manipulá-las posteriormente, conforme suas necessidades. Isso é possível porque a ferramenta possui integração direta a um banco de dados. Após a criação da ontologia, a ferramenta armazena-a no banco de dados no formato que vem sendo mais utilizado dentro da Web para representação de metadados na Web: o RDF da W3C, cuja sintaxe é feita em XML.

A ferramenta também oferece uma lista de consultas, com a qual os usuários podem consultar sua base de dados e buscar tanto por metainformações da ontologia, como: tipo, nome da ontologia, autor e outras informações que estejam no conteúdo da ontologia, ou seja, dentro do documento RDF propriamente dito.

2.5.8.4 Modelo utilizado

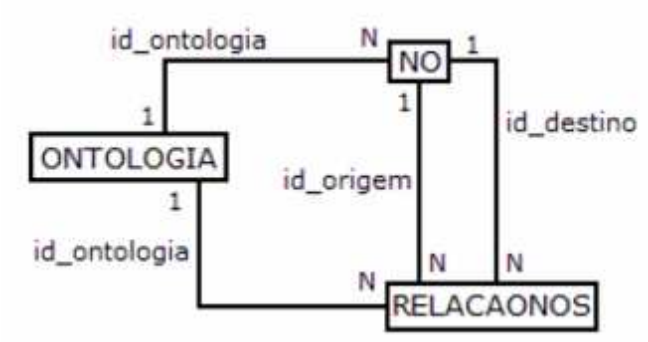

Figura 12: Modelo de dados do OntoEditor

Conforme mostra a figura 12, a camada de dados do OntoEditor é composta por três tabelas (Ontologia, Nó e Relação Nós), as quais armazenam as definições sobre as ontologias (VASCONCELOS, 2003).

2.5.8.5 Pontos fracos

O principal ponto fraco do OntoEditor é a "imaturidade", pois a tarefa específica da edição de ontologias ficou limitada, com poucas ações. O software ainda apresenta poucas funcionalidades, pois se trata de uma ferramenta em fase inicial de desenvolvimento. Basicamente, é uma ferramenta de prova de conceito.

2.5.9 ONTOSTUDIO

OntoStudio é implementado como uma extensão comercial na plataforma de desenvolvimento Eclipse, com vantagens como o conceito de plug-in. Disponível com um modelo de memória principal, ou banco de dados, é, portanto, escalável e adequado para a modelagem de grandes ontologias (ONTOPRISE, 2010).

2.5.9.1 Características

OntoStudio é um ambiente de modelagem mais ampla comercialmente para a criação e manutenção de ontologias. Distingue-se através de funções globais na modelagem da ontologia intuitiva. Também tem a capacidade de importar muitas estruturas, esquemas e modelos. Entre as funções mais importantes, estão: a ferramenta de mapeamento, a regra do editor gráfico e ambiente de teste integrado. Com a ferramenta de mapeamento, é possível mapear estruturas heterogêneas em si de forma rápida e intuitiva. O editor gráfico para o ambiente de teste integrado garante a qualidade da modelagem (ONTOPRISE, 2010).

2.5.9.2 Ponto forte

Além de a ferramenta comportar diversos formatos de arquivos, como OWL, RDF, RIF, também pode importar outras estruturas, como UML 2.0, esquemas de banco de dados (Oracle, MS-SQL, DB2, MySQL), tabelas do Excel, e-mails do Outlook e estruturas de pastas do sistema de arquivos.

2.5.9.3 Ponto fraco

Apesar de suportar a utilização de esquemas de banco de dados, o OntoStudio não possui integração direta com eles, ou seja, necessita de outros aplicativos para se comunicar com o banco de dados, como OntoBroker (parte central do sistema, responsável pelo processamento da ontologia e toda sua lógica), SemanticMiner (responsável pelo acesso à ontologia pelo OntoBroker e acesso aos dados corporativos, como documentos word, pdf, planilhas e banco de dados), SemanticGuide (parte deste conjunto de aplicativos para uma solução de repositório de conhecimento específico) e, por fim, o SemanticIntegrator (disponibiliza uma visão única de informação das heterogêneas fontes de informação corporativa, utilizando a ontologia como modelo semântico integrador) (ONTOPRISE, 2010). Vale ressaltar que os aplicativos pertencem a um mesmo grupo.

2.5.10 OWL NeOn ToolKit

2.5.10.1 Arquitetura

Software multiplataforma, desenvolvido em Java, o NeoOn Toolkit utiliza-se do framework do Eclipse para a criação de ontologias, gerando um arquivo owl. Este software não contém conexão com banco de dados.

Com a versão comercial é possível utilizar a integração com banco de dados. Na especificação não consta com qual banco de dados é possível se conectar e a partir de qual banco é possível fazer importação (NeOn ToolKit, 2010).

2.5.10.2 Características

Promove uma ferramenta de metodologia para suportar desenvolvimento e gerenciamento de ontologias.
Principais Características:

- Permite adicionar *plug-ins*
- Licença *open source* - o desenvolvedor pode continuar aprimorando o software
- Visualização da ontologia
- Visualização do código-fonte, taxonomia e adição de comentários.

2.5.10.3 Pontos fortes

O ponto forte desse sistema é a possibilidade do desenvolvedor programar e redefinir o código-fonte de acordo com sua necessidade.

2.5.10.4 Pontos fracos

Não possui integração direta com banco de dados e não fornece detalhes sobre como formar ontologias extraindo informações do banco de dados.

2.5.10.5 Utilização

A utilização desse software é bem simples, facilitando o entendimento para o usuário. O manual do usuário é completo, inclui dicas de como criar plug-in, classes, propriedades e todos os recursos que software provê. A figura 13 mostra a interface do software. É possível observar a navegação através das classes e a edição de suas propriedades (NeOn ToolKit, 2010).

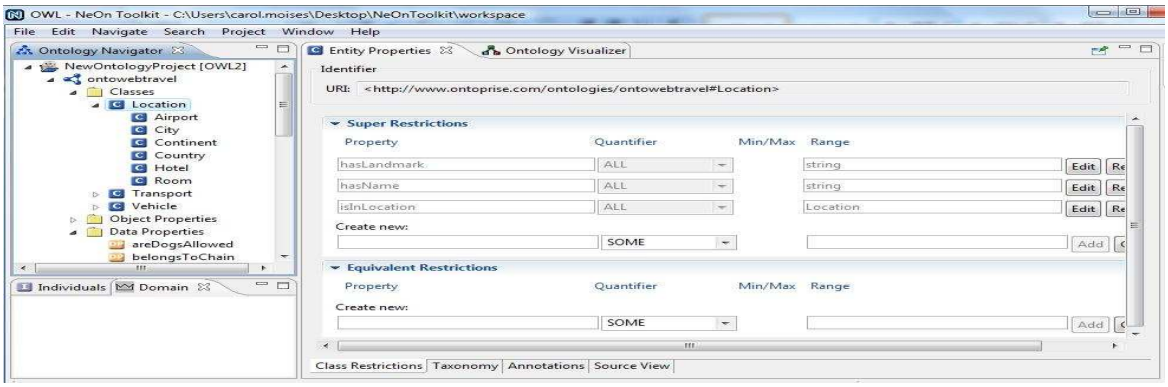

Figura 13: Propriedades de uma entidade

2.5.11 Protégé

2.5.11.1 Arquitetura

O Protégé permite a construção de ontologias de domínio, tornando possível a criação de bases de conhecimento guiadas por ontologias que representem o conhecimento. Seus módulos de visualização permitem a navegação por entre as classes do domínio e *plug-ins* auxiliam no desenvolvimento da ferramenta e acrescentam flexibilidade na manipulação das ontologias (PROTÉGÉ PROJECT, 2010).

Conforme pode ser visto na figura 14, o Protégé é desenvolvido em camadas, possuindo a capacidade de armazenar dados em bases de dados relacionais, bem como em arquivos de textos simples. Essa funcionalidade é realizada através da camada de armazenamento.

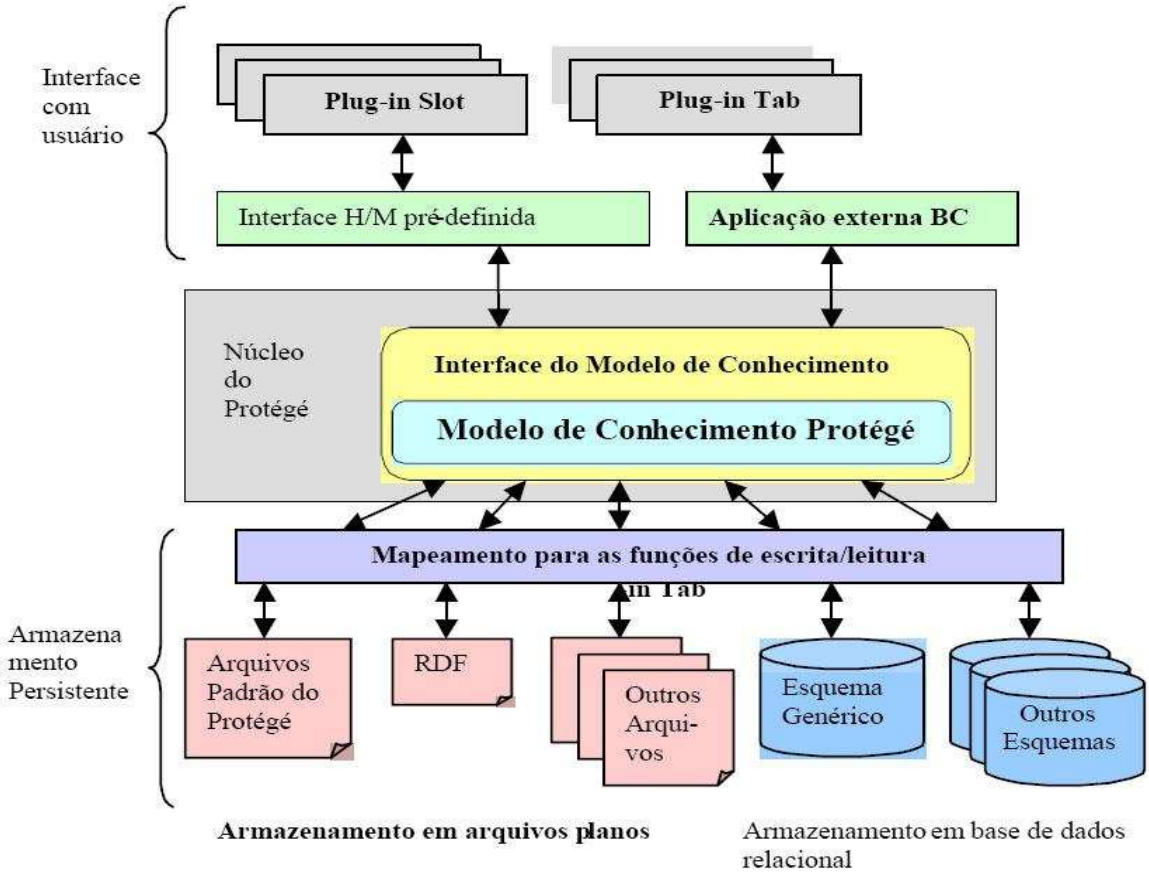

Figura 14: Arquitetura do Protégé

2.5.11.2 Características

Foi desenvolvido pelo Departamento de Informática Médica da Universidade de Stanford e em seu projeto original era uma ferramenta de aquisição de conhecimento limitada a um sistema especialista para oncologia. Sus desenvolvedores decidiram então compartilhar o código-fonte para alavancar as possibilidades da ferramenta.

Suporta os formatos:

- RDF/XML
- OWL/XML
- OWL Functional Syntax
- Manchester OWL Syntax
- OBO 1.2
- KRSS2 Syntax
- Latex
- Turtle

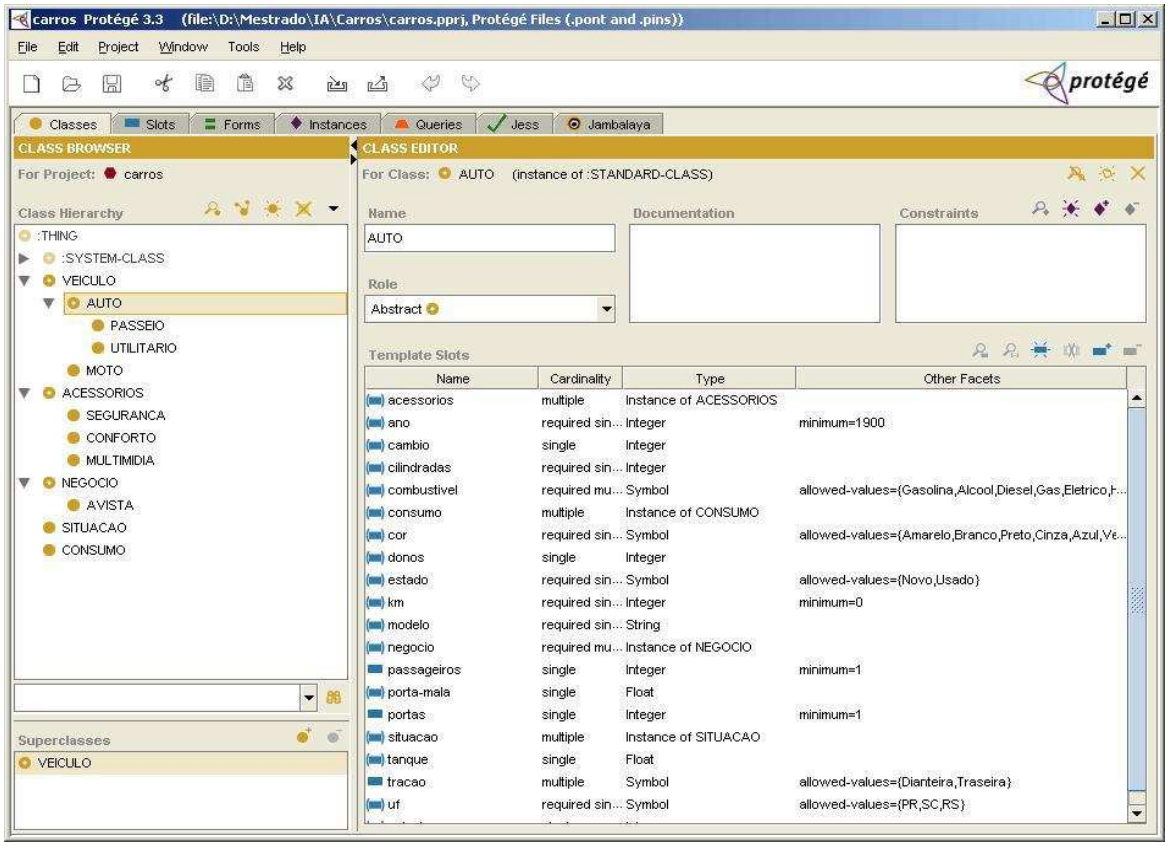

Figura 15: Exemplo de navegação entre as classes

Conforme pode ser visto na figura 15, sua interface gráfica provê acesso à barra de menus e barra de ferramentas, além de apresentar cinco áreas de visualização (views), que funcionam como módulos de navegação e edição de classes, atributos, formulários, instâncias e pesquisas na base de conhecimento, propiciando a entrada de dados e a recuperação das informações.

2.5.11.3 Pontos fortes

O Protégé é uma ferramenta Java de código aberto, com arquitetura extensível, para criação de aplicações baseadas em conhecimento, possibilitando então o ajuste para cada caso. Suporta vários formatos, dentre eles os mais utilizados, como OWL – Ontology Web Language.

Por ser uma ferramenta integrada, é utilizada por desenvolvedores de sistemas e especialistas para o desenvolvimento de Sistemas Baseados em Conhecimento.

2.5.12 RDote - Relational Databases to Ontology Transformation Engine

2.5.12.1 Arquitetura

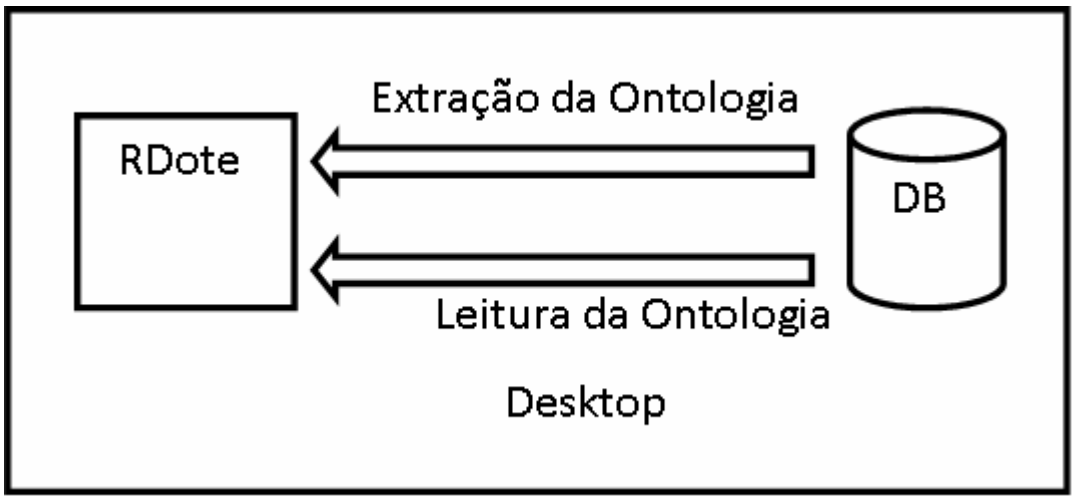

Figura 16: Arquitetura do RDote

O Rdote pode ser utilizado em domínios diversos, tais como bases de dados na área de construção do conhecimento - bases de artigos, bibliotecas, entre outros - conforme pode ser visto na figura 16.

Qualquer usuário com um mínimo de conhecimento na edição e construção de ontologias e que saiba os conceitos do domínio em questão poderá elaborar as inferências apropriadamente (RDOTE PROJECT, 2010).

2.5.12.2 Características

Ferramenta para edição de ontologias *desktop* que implementa a visualização arbórea e de grafos.

- Não depende de um software auxiliar, como um interpretador, sob o qual terá que ser executado
- Construído sobre a tecnologia JAVA, é independente de plataforma
- As funcionalidades do Rdote são baseadas no mapeamento das informações relacionais e das estruturas ontológicas
- As bases aceitas pelo programa são Oracle e MySQL.

O programa importa tanto projetos criados pelo próprio programa quanto ontologias armazenadas em bases de dados ou arquivos do tipo OWL (RDOTE PROJECT, 2010).

2.5.12.3 Pontos fortes

O RDote já vem com uma de base de testes e uma estrutura armazenada em arquivo OWL para a prova de conceito. Tem interface bem elaborada, com seções de união de consultas, assim como elaboração de consultas de inferência

2.5.12.4 Pontos fracos

Ausência de documentação do software e as dificuldades de manipulação da ferramenta na importação de projeto previamente salvo e na importação de arquivos OWL. Conclui-se que a ferramenta ainda encontra-se em fase de desenvolvimento, com muitas oportunidades de melhora.

2.5.13 Semantic Turkey

Semantic Turkey é uma ferramenta de código-fonte aberto destinada à organização semântica do conteúdo observado durante a navegação web. Ele está disponível sob a forma de um *plug-in* para o navegador Firefox. Substitui os marcadores tradicionais, inovando com a capacidade de anotar os dados do conhecimento e suas respectivas localizações (SEMANTIC TURKEY, 2010).

2.5.13.1 Características

O Semantic Turkey oferece a possibilidade de referenciar tanto as páginas como um todo, como também suas partes componentes. Possibilita o gerenciamento de projetos, pois cada usuário pode ter, de forma separada, suas anotações semânticas.

2.5.13.2 Utilização

Depois da instalação do *plug-in* no Firefox, é recomendado iniciar um projeto definindo como identificador uma URI.

Tendo definido uma classe dentro da estrutura que representa a ontologia, o usuário pode selecionar e arrastar para dentro dessa classe um recorte de texto ou um endereço web. A ontologia é preenchida através da funcionalidade do “drag&drop”. Sendo assim, as informações que compõem a ontologia são selecionadas nas páginas web e arrastadas para dentro da estrutura da ontologia. O usuário pode utilizar uma estrutura que já esteja pronta para preenchê-la, pode construir sua própria ontologia ou, ainda, pode utilizar uma ontologia existente e incrementá-la em sua estrutura ou informação (SEMANTIC TURKEY, 2010).

2.5.13.3 Pontos fortes

Ferramenta que apresenta código aberto e que trata a semântica web, com camada de dados, aplicação e domínio bem definidos, além de ser orientada a objeto.

2.5.13.4 Pontos fracos

Não oferece nenhuma opção para banco de dados, o que poderia ser uma melhoria a ser considerada.

2.5.13.5 Recursos

O Semantic Turkey tem recurso para a construção da estrutura de classes que compõe a ontologia, para o preenchimento desta estrutura com informações, para a visualização gráfica e para a busca por informações dentro da ontologia.

2.5.13.6 Arquitetura

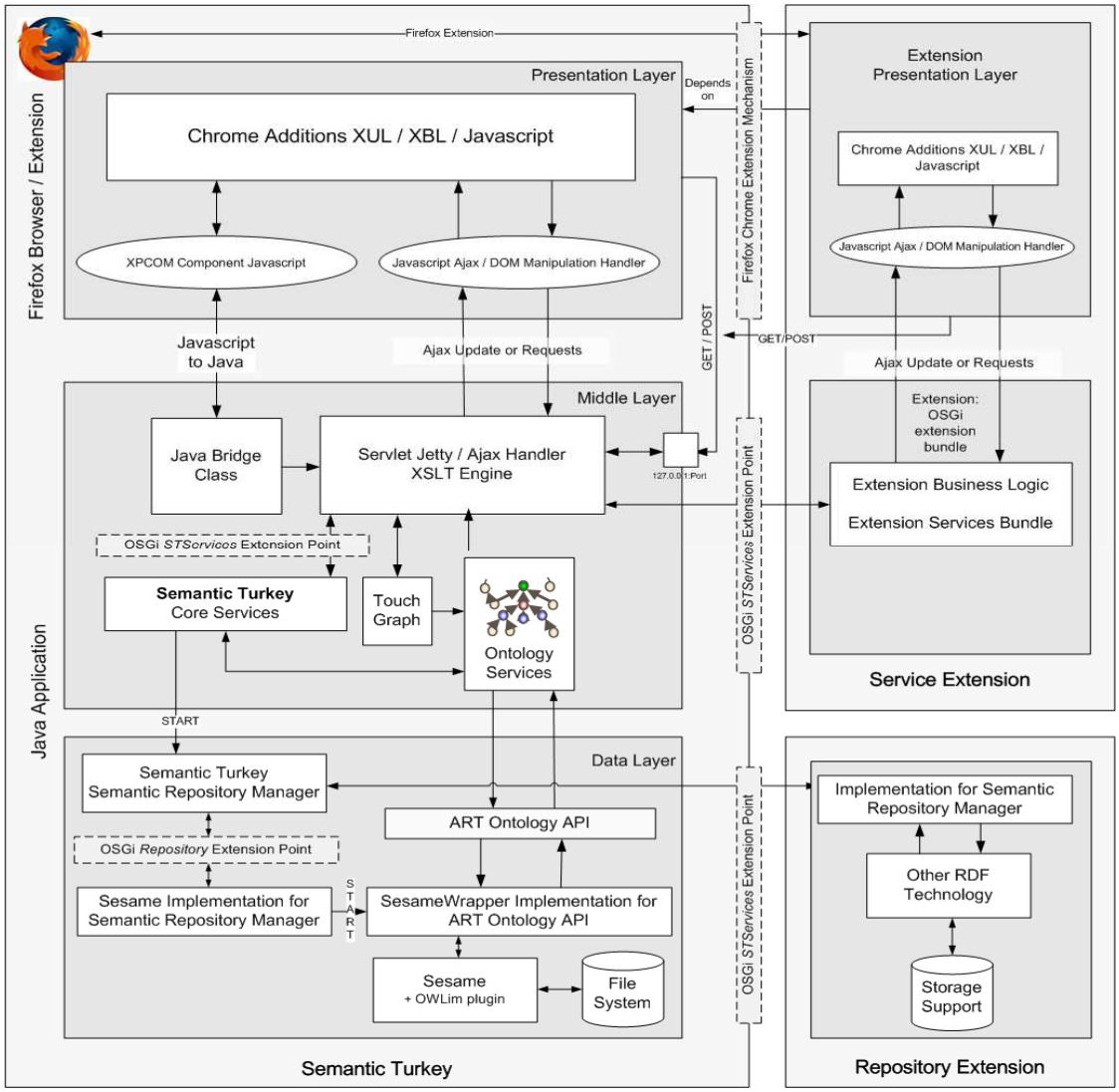

Figura 17: Arquitetura do Semantic Turkey

A arquitetura do Semantic Turkey é melhor visualizada na figura 17. Ela é composta por uma camada de apresentação, que é construída através da API de extensões do Firefox. Possui uma camada de negócio, que é construída em Java. Possui também a camada onde os dados são armazenados.

A camada de dados é representada principalmente por um componente de acesso e gestão da ontologia RDF/ OWL. Este componente consulta e gerencia ontologias através de uma API específica (ART Ontologia API), que fornece uma abstração simples sobre as tecnologias RDF/OWL.

A camada de aplicação contém ontologias (e dados relacionados) necessários para o pedido de coordenar e organizar os serviços. Essas ontologias devem permanecer

ocultas para o usuário. No entanto, em muitos casos, o seu conteúdo é implicitamente oferecido ao usuário através de funcionalidades disponíveis.

A camada de domínio inclui todo o conhecimento de domínio que é diretamente manipulado pelo usuário, e inclui dados de conhecimento importados da web, dados pessoais e informações anotadas, definidos a partir de páginas web (SEMANTIC TURKEY, 2010).

2.5.14 TopBraid Composer

TopBraid é uma coleção de soluções integradas para web semântica, visualizada na figura 18. Todos os componentes funcionam em multiplataforma e implementam os padrões W3C (TOPBRAID COMPOSER, 2010).

2.5.14.1 Arquitetura

Tem suporte para UML, XML, databases, RDF, OWL e SPARQL. Utiliza o framework do Eclipse, API Jena e é desenvolvido em JAVA, conforme apresentado na figura 18.

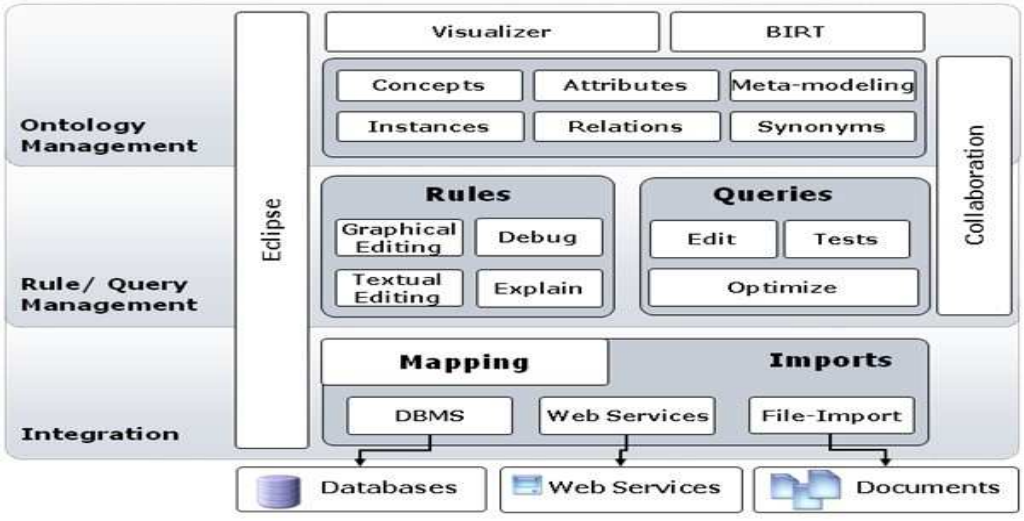

Figura 18: Arquitetura do TopBraid

Existem três versões para o TopBraid, sendo que duas são pagas:

- Free Edition (FE) é uma versão introdutória, somente com as principais funcionalidades.

- Standard Edition (SE) inclui todas as funcionalidades da versão livre, mais visualização de gráficos, importação, suporte avançado, dentre outras coisas.

- Maestro Edition (ME) inclui todas as funcionalidades da versão *standard*, mais suporte à versão TopBraid Live e Ensemble, como também SPARQLMotion e outras ferramentas.

2.2.14.1 Características

Diversas formas de importação de ontologias disponíveis: XML, RDF e informações de banco de dados relacionais. O que chama atenção no software é a variedade de visualizações que ele fornece: diagramas de classe, código-fonte, mapas geográficos, visualização de instrução SPARQL e de classes no modelo hierárquico (TOPBRAID COMPOSER, 2010).

2.2.14.2 Pontos fortes

A aplicação oferece uma ajuda dinâmica. A cada evento do software, a ajuda dinâmica é disparada. Por exemplo, ao abrir uma classe, a ajuda dinâmica fornecerá passos de como criar, editar ou excluir uma classe. O SPARQL é uma linguagem poderosa, com a qual é possível criar filtros de pesquisa para buscar por inferências. Existe a possibilidade de verificar a sintaxe e também de *debug* dos comandos de execução SQL. SPARSQL é um dos padrões da W3C para pesquisa RDF. A notação é similar ao SQL utilizado nos bancos de dados relacionais, porém contém um formato diferente exclusivo para ler *tags* do RDF. A integração com o Google Maps também é interessante, pois é possível utilizar uma ontologia de países, criar a consulta com o SPARQL e visualizar a região com o Google Maps (TOPBRAID COMPOSER, 2010).

2.2.14.3 Pontos fracos

O software no geral é rico em funcionalidades. No entanto, o Report Design exige um esforço por parte do usuário, pois a documentação não abrange detalhadamente como criar um relatório, como conectar um banco de dados ou como efetuar a importação. Os recursos mais atraentes do Software, como BIRD, SPARQLMotion e Visões Geográficas, funcionam apenas na versão Master.

2.2.14.4 Recursos

Existem vários recursos para este software. Os mais importantes estão listados abaixo:

Diagrama de Classe – UML – gera automaticamente diagrama de classes para as definições de classes. Também pode exportar o diagrama para HTML.

Inferência e verificação de consistência – suporte a vários mecanismos de verificação de consistência. As ferramentas OWLIM, Jena, Pellet e SPARQ Notação de Inferência (SPIN) aumentam a desempenho de pesquisas. Várias formas de inferência podem ser combinadas facilmente.

Base para Regras – o sistema permite a execução e edição de regras no SPARQL, SWRL, Jena e regras do Oracle.

SPARQL Queries – interface para edição de SPARQL com possibilidade de *debug* e verificação de sintaxe. As buscas podem ser executadas no modelo básico ou no modelo com regras, utilizando dados RDF ou banco de dados relacionais.

Mapeamento de Fonte de Dados – interface para mapeamento semântico de modelos utilizando OWL, axiomas, SWRL regras e SPARQL Construção de instrução. Os mapeamentos podem ser visualizados no diagrama de classe e executados de várias formas.

Geografia e mapeamento de localização – utilizado para conectar fontes RDF ou ontologias com geoespacial. Coordenadas geográficas podem ser visualizadas incluindo-se a interface Google Maps. A aplicação pode ser usada para pesquisar informações de uma região especifica. Pode ser baseada em SPARQL ou inferência.

Calendário e gráficos Mash Ups – Se uma ontologia contém data e hora, então as datas podem ser visualizadas como Calendário. Outros valores numéricos podem ser vistos e analisados através de gráfico linear.

2.2.14.5 Utilização

O software tem varias funcionalidades, sendo uma delas o SPARQL

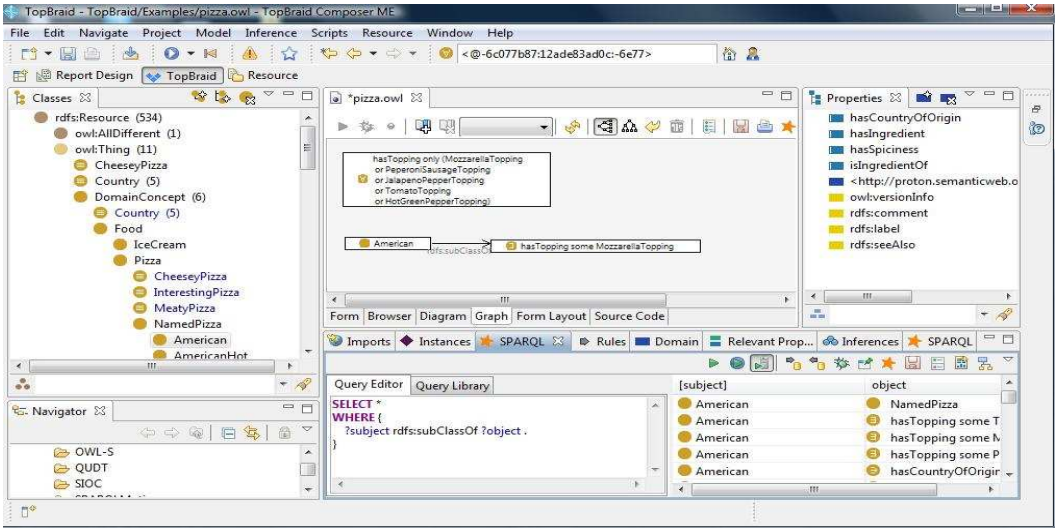

Figura 19: Busca por inferência

A figura 19 demonstra a tela de busca por inferência. Selecionando um domínio e clicando no botão play, a aplicação buscará por inferências.

2.5.15 WEBODE

O WebODE é uma ferramenta para modelagem do conhecimento, baseada nos princípios da Methontology (WEBODE, 2010). Os conceitos aplicados pelo Webode são:

- Conceitos são identificados pelo seu nome, que podem ter sinônimos e abreviações. A linguagem natural descritiva pode ser incluída.

- Atributos de classes são atributos que especificam características de uma classe e quais valores são os mesmos para todas as instâncias do conceito. Elas são definidas com a seguinte informação: nome do atributo (que precisa ser diferente dos demais com mesmo conceito), nome do conceito pertence a (atributos são locais aos conceitos, que são dois diferentes conceitos que podem ter diferentes atributos com o mesmo nome), tipos de valores ou faixa, que podem ser tipos básicos de dados (String, Integer, Cardinal, float) ou ainda conceitos (especificados pelo nome do conceito), cardinalidade mínima e máxima, que é o número de restrições dos valores que a classe de atributos pode ter.

- Instância de atributos: são atributos nos quais os valores podem ser diferentes para cada instância do conceito. Ele tem as mesmas propriedades dos atributos de classes e duas propriedades adicionais, valor mínimo e valor máximo, cada um usando atributos com valores do tipo numérico. Valores inseridos nos atributos de instância são interpretados como valores-padrão deles.

- Grupos de conceito são instanciados separando-se conceitos que são apenas conhecidos como partições. São usados para criar, separar e particionar exaustivamente as classes.

2.5.15.1 Arquitetura

WebODE foi construído de acordo com quatro camadas: Cliente, Apresentação, Negócio, Lógica e camadas de banco de dados. Em todas essas camadas, existe uma tecnologia padronizada. A camada de apresentação foi feita utilizando-se JAVA Servlets e JSP. A camada de negócio usa JAVA e RMI-IIOP. Finalmente, a camada de banco de dados usa JDBC e Oracle (WEBODE, 2010).

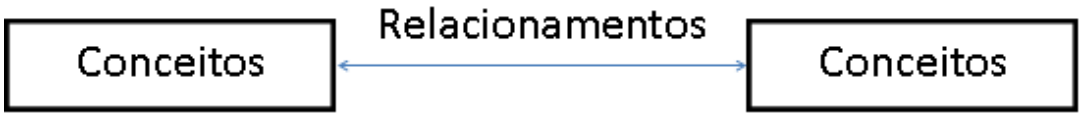

Figura 20: Relacionamento de conceitos - WebODE

Construções e relações são relacionamentos pré-definidos no modelo de conhecimento Webode. Isto é, funcionam com o estado bruto da ontologia, conceitos interligados por relacionamentos a outros conceitos, conforme mostra a figura 20.

2.5.15.2 Características

- O Framework Webode foi desenvolvido no contexto da necessidade de integrar a engenharia ontológica de banco de dados, suportando três grupos de atividades

- Desenvolvimento ontológico, gerenciamento e atividades de população

- Serviços de *middleware* ontológico para permitir o fácil uso e integração de tecnologias ontológicas na informação de sistemas

- Aplicações de conjuntos de bases ontológicas de desenvolvimento para facilitar a criação da Ontologia de aplicativos.

2.5.15.3 Pontos fortes

- É um framework escalonável
- Possui API que facilita a estruturação de uma ontologia através da linguagem de programação JAVA
- Permite a edição colaborativa de ontologias
- Como o WEBODE é um framework, pode ser utilizado em aplicações do mundo real, pois dispõe de um *middleware* para integração de ontologias
- Qualquer engenheiro de ontologias que tenha conhecimento da linguagem de programação JAVA pode implantar um módulo na linguagem
- Para a utilização do framework, basta um ambiente de desenvolvimento JAVA para sua implantação.

2.5.15.4 Pontos fracos

- Por ser um framework, suas funções são muito genéricas e seria necessária uma implantação mais profunda para atingir determinados domínios.

2.5.15.5 Utilização

Pelo motivo de o Webode ser um framework, apenas foram levantadas algumas das funções de sua API, Abaixo, algumas das mais importantes.

- Criação de ontologia
- Inserção de termos na ontologia criada
- Capacidade de referenciar outra ontologia
- Inserção de relações entre os termos
- Inserção de atributos nos termos
- Atribuição de valores aos atributos.

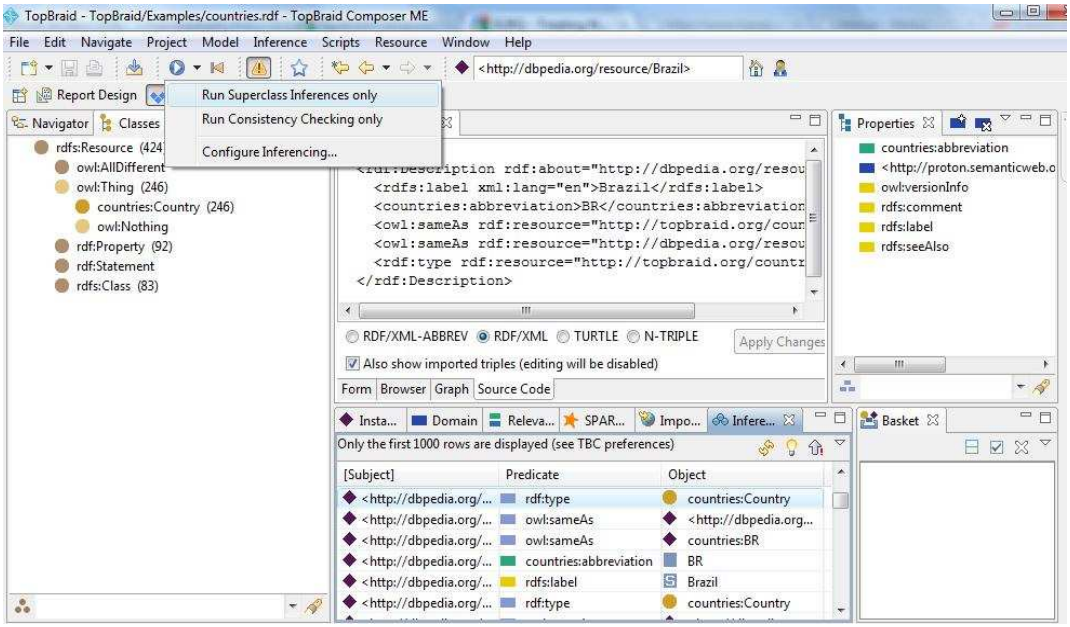

Figura 21: Funcionalidade de busca

Na figura 21, é possível observar a execução do WebODE, onde se nota a opção de manipulação de classes e de suas propriedades.

2.6 Trabalhos Correlatos

2.6.1 Veronto

O trabalho é uma proposta do uso das metapropriedades ontológicas apresentando as restrições sobre relacionamentos hierárquicos por elas impostas, para validação de modelos conceituais expressos por meio de diagramas de classe, buscando uma forma de relacionar racionalmente ontologia e modelagem conceitual (VILLELA, 2005).

A contribuição principal deste trabalho consiste em apresentar um mapeamento das classificações de propriedades apresentadas por GUARINO e WELTY (2000a), geradas a partir de combinações das metapropriedades, e das restrições que estas impõem sobre relacionamentos hierárquicos, nos elementos do diagrama de classes da UML. Espera-se que o uso destas restrições permita a construção de modelos conceituais mais manuteníveis e flexíveis. A aplicação desta metapropriedade aos elementos do domínio exige que seja realizada uma análise ontológica dos mesmos. Dessa forma, este trabalho propõe a introdução de uma etapa adicional no processo de desenvolvimento de software, que consiste na verificação do modelo conceitual a partir de uma análise ontológica dos elementos do domínio.

Este trabalho trata apenas da apresentação através do diagrama de classes e da restrição de sua utilização.

2.6.2 DERONTO

Propõe possibilitar que o aumento do uso de ontologias em aplicações variadas torne mais fácil aos profissionais de informática o aprendizado dos fundamentos teóricos dos métodos de construção e das linguagens de representação de ontologias. Neste trabalho, parte-se do princípio que o DER é uma ferramenta de modelagem de dados bastante difundida e utilizada pelos profissionais de informática. Se este conhecimento existente for reutilizado na construção de ontologias, então estas últimas serão construídas mais facilmente e utilizadas de forma mais ampla no desenvolvimento de aplicações. Assim, o objetivo da dissertação é apresentar um método, denominado DERONTO, para construção e representação formal de uma ontologia a partir de um DER (CALIARI, 2007).

O autor entende que a escolha do DER como ponto de partida é devido à popularidade deste modelo junto aos analistas de sistemas, programadores e administradores de bancos de dados, reduzindo o tempo de aprendizado do DERONTO. Outro fato é que, em muitos casos, o DER já está presente na documentação do banco de dados e, teoricamente, já foi validado pela área responsável e pode ser aproveitado, reduzindo o tempo de elaboração de uma ontologia. Possibilitar o compartilhamento e a interoperabilidade do conhecimento entre os domínios, estruturar o domínio de forma que se permita sua compreensão com maior clareza e objetividade e permitir a reutilização de conceitos são algumas das vantagens da ontologia em sistemas de informações.

Este trabalho apresenta apenas a criação de ontologias a partir do DER e não trata o modelo lógico ou físico.

2.7 Discussão

Neste capítulo, foram descritos trabalhos relacionados ao proposto nesta tese, considerando-se duas abordagens: as metodologias "clássicas", com modelos baseados em métodos formais para construção de ontologias e baseados em engenharia reversa.

Foram citadas as metodologias de engenharia reversa propostas por (SUGUMARAN *et al.*, 2002), (SAIAS *et al.*, 2003) e (DOGAN *et al.*, 2002), porque nelas procurou-se identificar aspectos semelhantes aos da tese proposta para enriquecer o método, embora o ponto de partida para receber informações dessas metodologias é um modelo entidade-relacionamento. Quando considerada a abordagem relacionada às metodologias de engenharia reversa baseadas na construção de ontologias, são descritas de forma sucinta.

Quando considerada a abordagem relacionada às metodologias clássicas, observa-se que a literatura pertinente é focada na descrição do problema para o qual a ontologia será construída, para só então verificar se a metodologia satisfaz os requisitos especificados para construção. A aplicabilidade depende da finalidade da ontologia, levando-se em conta o fato que é necessário um bom conhecimento por parte do projetista sobre o caso específico.

O processo de aquisição de conhecimento nas metodologias clássicas é uma atividade independente, mesmo que seja coincidente com outras atividades do desenvolvimento da ontologia. Possíveis fontes de conhecimento são projetistas, livros, manuais, figuras, tabelas e outras ontologias. Possíveis técnicas para levantar requisitos são *brainstorming*, entrevistas, análise de texto. As técnicas e as fontes podem ser combinadas. Por exemplo, a análise de texto em conjunto com entrevistas com projetistas pode ser usada para refinar um glossário de termos utilizado na metodologia *Methontology* (FERNÁNDEZ *et al.*, 1997).

A metodologia que mais se aproxima da utilização e tratamento do modelo conceitual é a Ontoclean (GUARINO e WELTY, 2002), que será utilizada no Capítulo 4 e implementada no Capítulo 5.

3. MINERAÇÃO DE DADOS

A origem do *data mining* está associada a análises estatísticas executadas em meados dos anos 60. Com o decorrer do tempo, as técnicas foram sendo aperfeiçoadas e mais áreas sendo abrangidas, surgindo os conceitos de Lógica Nebulosa (Fuzzy Logic), Redes Neurais Artificiais (Artificial Neural Networks) e Árvores de Decisão (Decision Tree). No início da década de 90, os pesquisadores o classificaram como uma área da Inteligência Artificial (CISTER 2005).

A necessidade surge quando se passa a constatar que se possui um grande número de dados, mas estes não trazem a informação devida, ou essa informação não está suficientemente satisfatória (HAN 2000).

Segundo COUTINHO (2003), "Data Mining (ou mineração de dados) é um processo para extrair informação válida, previamente desconhecida e de máxima abrangência, a partir de grandes bases de dados ou armazéns de dados, denominados *datawarehouse"*.

A tarefa de mineração de dados é feita seguindo os seguintes processos:

- **Limpeza**: são revisados o modelo e a base de dados para se remover dados irrelevantes ou que não estejam em um padrão;
- **Integração**: muitas vezes os dados encontram-se espalhados, em várias bases de dados (o que causa a falta de padronização deles) e em arquivos diversos;
- **Seleção**: definição do que realmente é importante e merece ser pesquisado;
- **Transformação:** os dados são levados a um padrão único;
- **Mineração:** aplicados algoritmos que seguem um metamodelo para a extração de dados;
- **Melhorias:** o metamodelo é reestruturado até se encontrar uma forma ótima para ele, abrangendo a maior quantidade, com maior qualidade de informação;

- **Apresentação do conhecimento:** os dados minerados são estruturados na forma a ser apresentada ao usuário final.

Sendo assim, cabe muito bem ao contexto proposto, pois se esperava a obtenção de dados não documentados que o sistema em operação não é capaz de fornecer a partir da análise, combinação, reestruturação e outras tarefas que sejam necessárias para a obtenção da informação.

3.1 Mineração de dados e ontologia

Existem estudos que visam aprimorar os conceitos utilizados por especialistas em mineração de dados com ontologia. No capítulo anterior foram apresentados editores e metodologias específicas que manipulam ontologias. Cabe agora apresentar alguns trabalhos que trazem mineração de dados e ontologias, que serão mostrados nas seções seguintes deste capítulo.

3.1.1 IDA

Este trabalho apresenta o conceito de assistente inteligente de descoberta (discovery assistants), que fornece aos usuários vários processos válidos de mineração de dados, a fim de que importantes opções não sejam potencialmente esquecidas, e faz um *ranking* desses processos válidos por diferentes critérios. É discutido também como é importante compartilhar conhecimento. A ferramenta IDA utiliza a ontologia para auxiliar o usuário na composição de processos válidos e úteis para mineração de dados. No protótipo, a ontologia contém, para cada operador:

- A especificação das condições sob as quais o operador pode ser aplicado, envolvendo um pré-requisito sobre o estado do processo de mineração, bem como sua compatibilidade com o operadore anterior
- A especificação dos efeitos do operador sobre o estado do processo de mineração e sobre os dados
- As estimativas dos efeitos do operador em atributos como velocidade, precisão, compreensibilidade do modelo, grupos lógicos, que podem ser usadas para restringir o conjunto de operadores a serem considerados em cada etapa do processo

- Esquemas pré-definidos para problemas genéricos, tais como mercado-alvo.

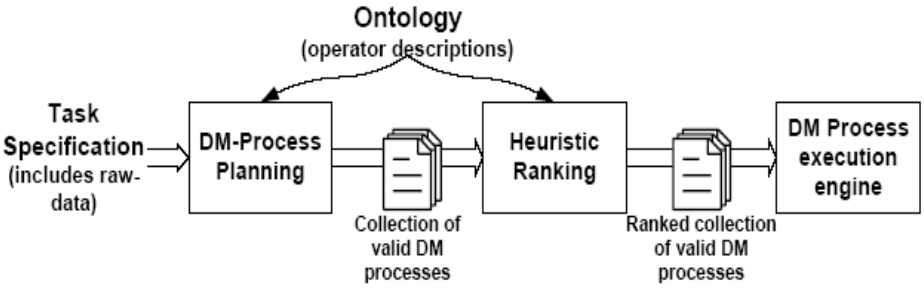

Figura 22: Processos do IDA

A ferramenta IDA (intelligente Discovery assistants) interage com o usuário para obter dados, metadados e metas. Em seguida, ela compõe o conjunto de processos válidos de acordo com as limitações implícitas pelas entradas do usuário, dados e ontologia. Essa composição envolve escolher o algoritmo de indução, módulos apropriados de pré-processamento e pós-processamento, bem como outros aspectos do processo, não considerados na exposição do trabalho. O IDA classifica os processos adequados para uma ordem sugerida com base em solicitações do usuário, e pode selecionar os planos da lista de sugestões, ajudando assim a classificação dos processos. Finalmente, o IDA produz código para executar e pode (automaticamente) sugerir processos sobre os dados selecionados, conforme demonstrado na figura 22.

As limitaçoes da ferramenta são: não produzir processos cíclicos e o gerador de código ainda não produzir um código de componentes mais complexos, tais como a seleção de recurso repetitivos ou amostragem progressiva. Ainda não estuda a produção de *ranking* com profundidade - por exemplo, o uso da correlação de Spearman, coeficiente de pesos, e o efeito nas posições ao longo de um *ranking*. No entanto, os processos perto do topo do *ranking* provavelmente seriam os críticos (especialmente para grande número de planos de processo gerado). (BRAZDIL, SOARES E COSTA, 2001). Este trabalho foi aplicado na base do KDD (Knowledge Data Discovery), na competição de 1998, e mostra resultados na utilização da ontologia como fator preponderante na definição e aprimoramento do aprendizado nos processos de mineração de dados.

3.1.2 Assistência à mineração inteligente de dados com o auxílio de ontologia: combinando o conhecimento declarativo e processual.

O trabalho apresenta uma combinação sinérgica de uma ontologia voltada à mineração de dados e utiliza o paradigma de raciocínio baseado em casos (CHAREST ET AL, 2006). Como ilustrado na figura 23, o assistente consiste basicamente em seis grandes componentes: uma base de casos para mineração de dados , uma ontologia para mineração de dados, um processo cognitivo, de regras, um raciocinador para descrição lógica e uma interface para mineração de dados. O assistente captura o conhecimento sob a forma de uma série de fases bem definidas, como as tarefas e atividades e, em seguida, define uma representação do caso para a mineração de dados. A maioria dos índices foi obtida a partir de medidas usadas na área de caracterização de dados, isto é, foram utilizados estatística e o referencial teórico da informação (CASTLIELLO, CASTELLANO e FANELLI, 2005). Para o componente de raciocínio do sistema CBR (resultados baseados em casos), foi implementado um algoritmo de aprendizado baseado em instância (AHA, KIBLER e ALBERT M , 1991). A avaliação preliminar da resolução baseada em caso produziu alguns resultados ao assistente de mineração de dados. Através da sua utilização, o sistema de CBR é capaz de aprender sobre os negócios e disponibilizar conhecimento para auxiliar um leigo em mineração de dados. A ontologia de mineração de dados foi utilizada formalmente por conceitos na captura de relacionamentos, restrições e regras e é capaz de complementar o sistema de CBR, ajudando um leigo em mineração de dados por meio de recomendações e heurísticas no decurso de um tarefa de mineração de dados. A implementação da ontologia possui duas fases distintas: (1) o alto nível de representação do conhecimento da metodologia CRISP-DM; e (2) a representação do conhecimento detalhado na mineração de dados na forma de conceitos e regras. Desta forma, um conjunto inicial de regras foi criado para a assistência de mineração de dados, ou seja, heurística, recomendações, respostas automáticas e compreensão dos dados. Para isto, foi utilizada a semântica do Protégé (O'CONNOR ET AL, 2005).

Posteriormente, um raciocinador baseado em regras (JESS) opera em um conjunto de Fatos. Os fatos consistem no fornecimento pelo usuário de atributos do caso. O sistema usa o termo de conselhos para representar toda a assistência fornecida

ao usuário como mensagem de texto, em que é feita uma clara distinção entre uma recomendação e uma heurística

A recomendação é um tipo mais formal de aconselhamento, definida como afirmação, enquanto a heurística deve ser interpretada menos formalmente por um usuário. O raciocionador de base de casos e ontologia tem bem definidas as funções de representação do conhecimento. A ontologia define e a mineração de dados gerencia os conceitos, tarefas, tipos de atividade e algoritmos, enquanto o raciocínio baseado em casos detém os passos de preparação e os parâmetros do modelo. A ontologia para mineração fornece assistência adicional, complementa o que a CBR não tem conhecimento para um usuário durante as várias fases do processo de mineração de dados.

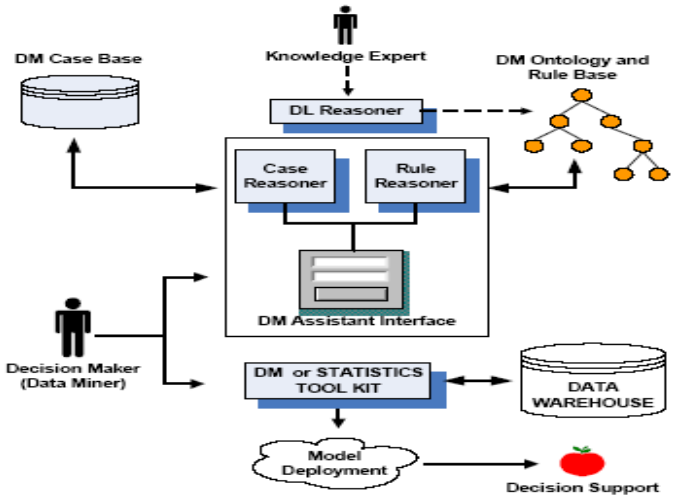

Figura 23: Arquitetura do sistema inteligente para mineração

O auxílio da ontologia para mineração de dados é principalmente restrita às três fases mais importantes, que são: compreensão, preparação e modelagem de dados. Embora o paradigma do CBR forneça o benefício da recuperação de casos similares, a parte da solução necessária é raramente uma correspondência exata para a mineração de dados. Assim, após as fases de recuperação e reutilização serem concluídas, o usuário examina os conteúdos escolhidos dos casos e faz a revisão de certos atributos, verificando a qualidade, preparação de dados e modelagem de dados, a fim de equipar o caso para refletir o estado do problema atual de mineração de dados.

A limitação do assistente é não prever a captura de todo o conhecimento necessário à mineração de dados para dar suporte a usuários em todas as possíveis circunstâncias. O assistente não detalha a ontologia do conhecimento obtido e é condicionada ao suporte de dados na fase de preparação para manipular com a qualidade de dados e requisitos de entrada do modelo. O conhecimento está disponível apenas para o ambiente Weka. Assim, os tópicos mais avançados, tais como meta-aprendizagem, seleção de recursos, grandes conjuntos de dados, comparação de modelos, métodos e os detalhes dos parâmetros de classificação ainda não são cobertos.

3.1.3 ONTO4AR: quadro para mineração com associação de regras

Este trabalho está baseado no uso de ontologias para a representação e introdução de domínio de conhecimento no processo de mineração. Ao definir as restrições com base em uma ontologia, a estrutura oferece um ambiente de mineração, independente do domínio do problema. Com esta simplificação na definição e no uso de restrições, reduzem-se as diferença entre regras descobertas e expectativas dos usuários.

É proposto um novo quadro para descobrir regras de associação. Essa estrutura oferece um ambiente para especificar restrições que permitem ao usuário controlar o processo de mineração. Mas, em vez de usar um linguagem específica para introduzir as limitações, ele utiliza uma ontologia para representar o conhecimento existente. Além disso, o usuário pode escolher o tipo de restrição a ser aplicado, tanto à redução do número de padrões descobertos e quanto a seu alcance. As restrições podem ser escolhidas em um conjunto de entidades pré-definido, com base nas relações entre os conceitos expressos na ontologia. Desta forma, o quadro oferece um ambiente universal de regras de associação de mineração, uma vez que é capaz de representar conhecimento básico em qualquer domínio, e fornece um conjunto de parâmetros e restrições, definidos independentemente do contexto do problema. Além disso, o quadro permite a definição de um novo tipo de restrições. Em particular, é possível introduzir restrições de outros conteúdos, uma vez que eles são definidos sobre a estrutura das ontologias.

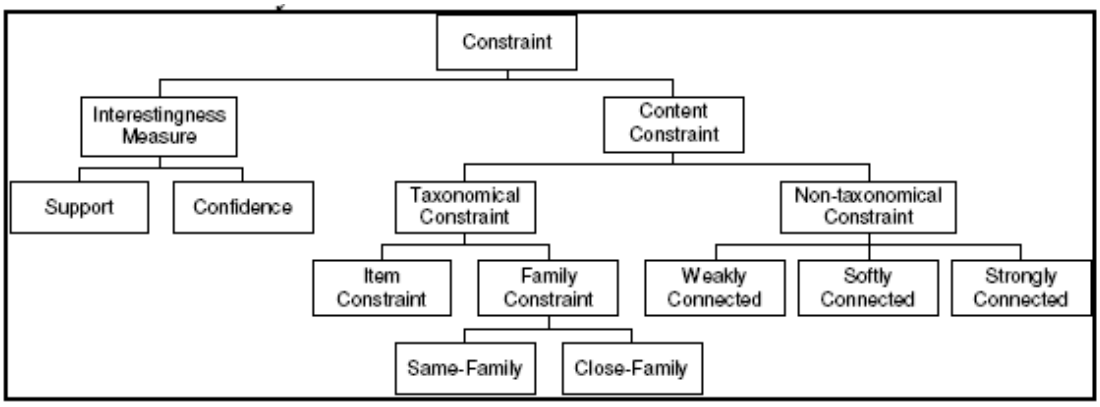

Figura 24: Categorização das restrições propostas

Através da ontologia, é criada uma taxonomia de restrições com o uso de medidas e conteúdo. As restrições de medidas derivam suporte e confidência, e as de conteúdo se subdividem em taxonômica e não-taxonômica. A taxonômica se subdivide em item e família; por sua vez, família se subdivide em famílias iguais e diferentes; enquanto a não-taxonômica se subdivide em fraca, suave e forte, como apresentado na figura 24.

O Onto4AR usa apenas a correspondência entre exemplos e conceitos, e a estrutura da ontologia e relações entre conceitos (axiomas e as relações de instância não são utilizados). Desta forma, a ontologia quase corresponde a um grafo acíclico.

O trabalho não oferece resultados de aplicação prática na apresentação de um modelo de dados e o tratamento específico que foi efetuado para a comprovação da aplicação de mineração propriamente dita.

3.1.4 Discussão

Um dos grandes problemas relacionados ao uso de conhecimento é a dificuldade para adquiri-lo de forma eficaz, pois muita investigação deve ser feita para elucidar um cenário de aplicação e definição das bases a serem pesquisadas, que geralmente têm dificuldades em expressar seus conhecimentos, pela falta de formalismo. Com os recentes desenvolvimentos na área de gestão de conhecimento, é possível fazer a representação do conhecimento com ontologias. Foram apresentados nas seções anteriores alguns trabalhos efetuados na área da mineração de dados com o auxílio de ontologia. Entretanto, esta utilização denota grande dificuldade de compreensão de mapear os passos e tornar compreensível o conhecimento, visto que a aplicação apenas

se dá em processos e na preparação dos arquivos em que serão aplicadas técnicas de mineração de dados. Nenhum trabalho faz referência à modelagem de dados para definição dos processos ou mesmo para a definir o modelo de mineração com a ontologia. Este trabalho propõe esta compreensão dos modelos para a criação de um modelo conceitual com auxílio da ontologia para uma possível mineração de dados, cuja proposta é apresentada no Capítulo 4 e com implementação no Capítulo 5.

4 PROPOSTA DE METODOLOGIA PARA A CONSTRUÇÃO DE MODELOS CONCEITUAIS COM O AUXÍLIO DA ONTOLOGIA

A ontologia de domínio pode ser utilizada como ferramenta que auxilia a análise de modelos de dados, uma vez que revela compreensão sobre a natureza dos objetos do domínio a serem estudados. Um estudo ontológico pode ser feito em uma etapa anterior ou posterior à criação do modelo de dados do domínio, para ser usado então em sua construção ou validação, respectivamente.

Verificou-se que as metapropriedades de *identidade, rigidez, unidade e dependência*, detalhadas na seção 2.3, impõem uma série de restrições sobre os relacionamentos entre conceitos em uma ontologia, principalmente no que diz respeito aos relacionamentos taxonômicos (GUARINO e WELTY, 2000a, 2000b, 2000c, 2000d, 2001; WELTY e GUARINO, 2001). Dessa forma, constata-se que tais restrições também podem ser úteis no estabelecimento de relacionamentos correspondentes a generalizações / especializações, em modelagem conceitual.

A literatura apresenta vários trabalhos que fazem uso das referidas metapropriedades, bem como as restrições por elas impostas, como base para uma análise detalhada dos elementos em modelos conceituais expressos na forma de modelo de dados. Desta forma, procura-se desenvolver uma metodologia para utilização de tais metapropriedades na validação de modelos conceituais de dados.

A metodologia desenvolveu-se em dois passos, que são detalhados nas seções 4.1 e 4.2. A técnica foi utilizada da seguinte maneira: primeiramente, foi realizado um mapeamento dos tipos de propriedade estabelecidos por GUARINO e WELTY (2000), vistos na Tabela 2, onde são apresentadox os oito tipos de propriedades, juntamente com a combinação das metapropriedades. Esta associação possibilita a aplicação das restrições sobre relacionamentos hierárquicos, impostas pelas metapropriedades ontológicas em questão aos elementos do referido diagrama. Posteriormente, foram

adicionadas regras para o uso de relacionamentos, preconizadas nos estudos ontológicos formais de WAND ET al. (1999), como uma forma de complementar a proposta, para tratar de outros tipos de relacionamentos, além dos hierárquicos.

A aplicação da metodologia deve permitir a obtenção de modelos conceituais mais próximos de um modelo específico de um domínio, ou seja, aqueles conceitos rígidos que deverão ser sempre modelados.

4.1 Mapeamento das classificações de propriedades em elementos do modelo de dados

Um estudo teórico das metapropriedades, bem como das regras que elas impõem aos relacionamentos hierárquicos entre os elementos do domínio, foi realizado. Foi definida então uma maneira de associar tais metapropriedades, por meio dos tipos de propriedades colocados por GUARINO e WELTY (2000a), aos elementos que compõem a modelagem conceitual, que podem variar em função do modelo utilizado.

As metapropriedades relacionadas à noção de *unidade* não interferem na classificação dos tipos de propriedades, como pode ser percebido na Tabela 2. Dessa forma, tais metapropriedades impõem restrições gerais sobre relacionamentos hierárquicos correspondentes a quaisquer tipos de propriedade, como segue: se tais entidades executam uma condição de unidade (+U), ou seja, constituem-se inteiros, elas não podem ser subclasses de classe que possui antiunidade (~U), ou seja, cujas instâncias não se constituem inteiros intrínsecos. Se as referidas entidades não possuem unidade (-U), ou seja, não executam uma condição de unidade comum para todas as suas instâncias ou estas não são inteiros intrínsecos, elas não podem ser subentidades de uma entidade que executa uma condição de unidade (+U).

A seguir, será detalhado o mapeamento dos tipos de propriedades em elementos do modelo de dados.

4.1.1 Mapeamento da classificação de propriedade “tipo”

A classificação de propriedade *Tipo* (GUARINO e WELTY, 2000a) fornece CI (+O), executa CI (+I) e é rígido (+R), conforme visto na Tabela 2. Assim, de acordo com o estudo sobre ontologias e modelagem conceitual realizado, conclui-se que o modelador das entidades correspondente a tal elemento seria uma *entidade forte*, uma vez que esta possui instâncias que podem ser identificadas.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a um *Tipo* somente pode ser uma entidade de nível superior que execute CI para suas instâncias (GUARINO e WELTY, 2000a), uma vez que sua identidade é herdada por suas subentidades

Outra restrição é que este tipo de entidade somente pode ser subentidade de entidades correspondentes a *Categorias*, *Tipos* ou *Quase-Tipos*, pois estas são as únicas que representam propriedades rígidas, que podem, por sua vez, subjugar outra propriedade rígida, exemplo visto na seção 2.4.9 na descrição da metodologia Ontoclean.

De acordo com os Princípios de Individualização Ordenável e de Expansibilidade Ordenável (LOWE, 1989), todos os elementos contidos em quaisquer domínios devem ser instâncias, direta ou indiretamente, de uma entidade referente a um *Tipo*.

4.1.2 Mapeamento da classificação de propriedade “quase-tipo”

A classificação de propriedade *Quase-Tipo* (GUARINO e WELTY, 2000a) não fornece CI (-O), executa CI (+I) e é rígido (+R), conforme visto na Tabela 1. Assim, conclui-se que o gerador de modelo de dados correspondente a tal elemento também seria uma *entidade forte*, uma vez que esta possui instâncias que podem ser identificadas.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a um *Quase-Tipo* também somente pode ser entidade de nível superior de entidades que executem CI para suas instâncias, uma vez que esta é herdada por suas entidades derivadas ou especializadas.

Outra restrição é que tal entidade deve ser necessariamente entidade derivada de,

no mínimo, uma entidade correspondente a um *Tipo*, com CI compatível, a fim de herdar a CI que executa.

A classe em questão somente pode ser entidade derivada de entidades correspondentes a *Categorias* e *Mixins*, que representam propriedades rígidas e semirrígidas, respectivamente. Porém, as classes deste último tipo não são recomendadas, segundo GUARINO e WELTY (2000a), devido à sua generalidade.

Exemplos correspondentes a *Quase-Tipos* são: Avaliação e Tipo_Avaliação, entidades que pertencem ao modelo conceitual de prescrição médica utilizado na implementação, pois cada uma delas possui instâncias que sempre as serão durante toda a sua existência, porém não podem ser identificadas globalmente, por uma característica que lhes é própria. A sua identificação é feita por meio de uma característica herdada de tipo_avaliação, que corresponde a um *Tipo*, conforme mostrado na figura 25.

Figura 25:Exemplo correspondente a um quase-tipo

4.1.3 Mapeamento da classificação de propriedade “papel material”

A classificação de propriedade *Papel Material* (GUARINO e WELTY, 2000a) não fornece CI (-O), executa CI (+I), é antirrígido (~R) e dependente (+D), conforme visto na Tabela 1. Assim, conclui-se que o elemento também seria uma *entidade forte*, uma vez que possui instâncias que podem ser identificadas.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a um *Papel Material* somente pode ser entidade de nível superior de entidades também correspondentes a *Papéis Materiais*, pois este tipo de propriedade é o único que, além de ser antirrígido, executa uma CI e é externamente dependente.

Outra restrição é que tal entidade deve ser necessariamente entidade derivada de,

no mínimo, uma entidade correspondente a um *Tipo*, com CI compatível, a fim de herdar a CI que executa.

Além de ser entidade derivada, direta ou indiretamente, de uma entidade referente a um *Tipo*, a classe em questão pode ser entidade derivada de entidades correspondentes a qualquer tipo de propriedade. Porém, GUARINO e WELTY (2000a) recomendam que tal entidade seja entidade derivada apenas de entidades que correspondam a papéis e propriedades rígidas.

Em um modelo de dados, este tipo de entidade representa um papel desempenhado pela sua entidade de nível superior em uma associação.

A modelagem efetuada desta forma é útil, uma vez que diminui o risco de inconsistências geradas como consequência do armazenamento de informações redundantes. Pois pode ocorrer que, dentro de um domínio específico, como por exemplo, em um hospital, existam médicos que sejam também pacientes. Assim, para que não seja necessário que informações de pessoas nesta situação apareçam duplicadas, tanto dentro da entidade médicos como na entidade paciente, aquelas informações básicas, que não variam, podem constar apenas uma vez dentro da entidade forte pessoa. Então, somente as informações específicas de médico e de paciente, respectivamente, ficariam armazenadas dentro da entidade derivada correspondente, demonstrada na figura 26.

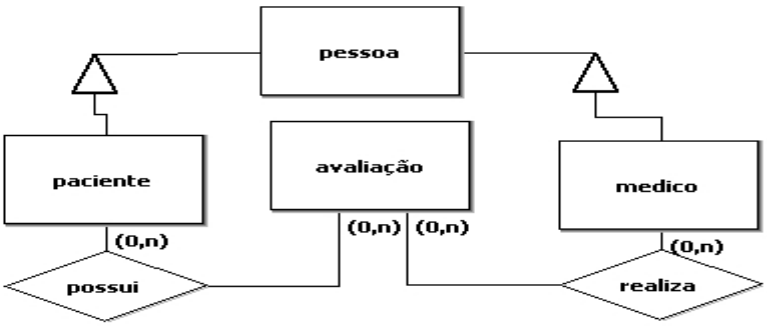

Figura 26: Exemplo correspondente a Papéis Materiais

4.1.4 Mapeamento da classificação de propriedade "sortal com fases"

A classificação de propriedade *Sortal com Fases* (GUARINO e WELTY, 2000a)

não fornece CI (-O), executa CI (+I), é antirrígido (~R) e não dependente (-D), conforme visto na Tabela 1. Assim, conclui-se que o gerador de modelo de dados correspondente a tal elemento também, neste caso, seria uma *entidade forte*, uma vez que possui instâncias que podem ser identificadas.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a um Sortal com Fases pode ser entidade de nível superior correspondente a quaisquer tipos de propriedades ordenáveis e não rígidas. Porém, GUARINO e WELTY (2000a) recomendam que sejam apenas outros Sortais com Fases ou Papéis Materiais.

A entidade em questão deve ser necessariamente entidade derivada de, no mínimo, uma entidade correspondente a um *Tipo*, com CI compatível, a fim de herdar a CI que executa.

Além de ser entidade derivada, direta ou indiretamente, de uma entidade referente a um *Tipo*, a entidade em questão pode ser entidade derivada de entidades correspondentes a qualquer tipo de propriedade não-dependente. Além disso, GUARINO e WELTY (2000a) recomendam que tal entidade seja entidade derivada apenas de entidades que correspondam a propriedades rígidas.

Este tipo de entidade representa uma fase pela qual passa, em um determinado momento, sua entidade de nível superior, correspondente a um *Tipo* ou *Quase-Tipo*, que, por sua vez, subjuga apenas entidades correspondentes a *Sortais com Fases*, representando todas as suas fases (GUARINO e WELTY, 2000a).

Exemplo de *Sortais com Fases* são: prescrição_atendimento e prescrição_solução, que, neste caso, representam fases pelas quais passa a entidade paciente, que corresponde à ilustrada na Figura 27.

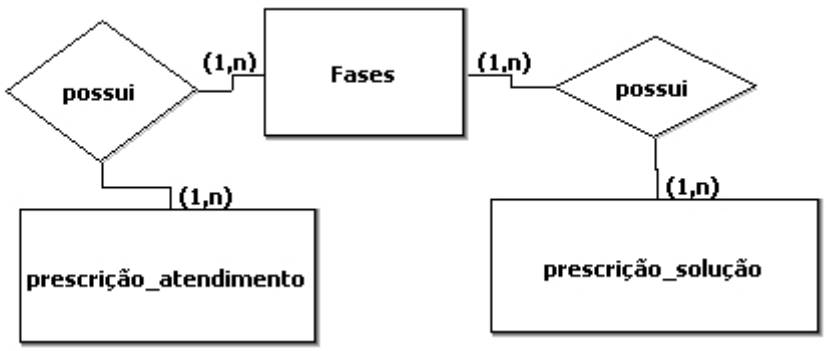

Figura 27: Exemplo de Sortais com Fases

4.1.5 Mapeamento da classificação de propriedade “mixin”

A classificação de propriedade *Mixin* (GUARINO e WELTY, 2000a) não fornece CI (-O), executa CI (+I) e é semirrígido (¬R), conforme visto na Tabela 2. Assim, conclui-se que o gerador de modelo de dados correspondente a tal elemento também neste caso seria uma *entidade forte*, uma vez que possui instâncias que podem ser identificadas.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a um *Mixin* pode ser entidade de nível superior de entidades correspondentes a quaisquer tipos de propriedades ordenáveis, pois transmite, por herança, sua CI. Normalmente, entidade correspondente a *Mixin* consiste em entidade de nível superior geral referente a propriedades rígidas e não-rígidas. Entretanto, GUARINO e WELTY (2000a) recomendam que *Mixins* não subjuguem propriedades rígidas e, devido à sua generalidade, até mesmo desencorajam seu uso.

A entidade em questão pode ser entidade derivada de entidades de quaisquer tipos, porém deve ser necessariamente entidade derivada de, no mínimo, uma entidade correspondente a um *Tipo*, com CI compatível, a fim de herdar a CI que executa.

A entidade prescrição_médica ou medicamento é um exemplo de Mixin_que vincula as entidades prescrição_médica e medicamento, que correspondem a *Tipo* e *Papel Material*, respectivamente. Como se pode perceber, a entidade atende a muitos vínculos, parecendo não adicionar informação importante ilustrada na Figura 28.

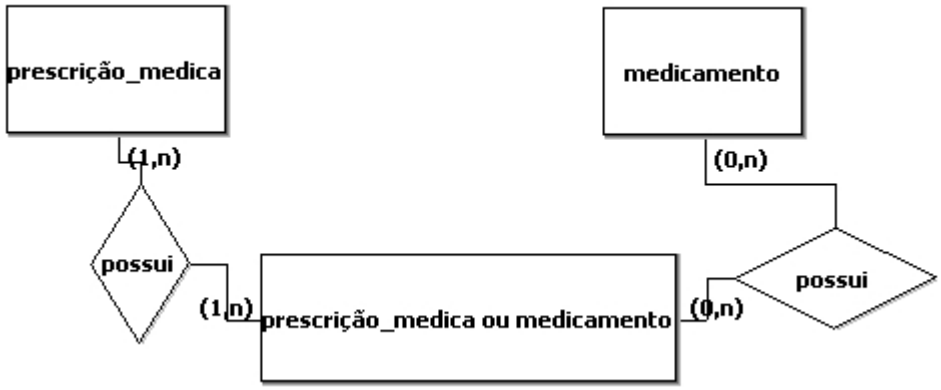

Figura 28: Exemplo de Mixin

4.1.6 Mapeamento da classificação de propriedade "categoria"

A classificação de propriedade *Categoria* (GUARINO e WELTY, 2000a) não fornece CI (-O), não executa CI (-I) e é rígida (+R), conforme visto na Tabela 4.1. Uma vez que não possui instâncias que podem ser identificadas, pois não executa CI, conclui-se que o gerador de modelo de dados correspondente a tal elemento seria uma *entidade fraca*.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a uma *Categoria* pode ser entidade de nível superior de entidades correspondentes a quaisquer tipos de propriedades. Normalmente, entidade correspondente a *Categoria* consiste em entidades de mais alto nível dentro de uma hierarquia, sendo utilizada para fins classificatórios.

A entidade Paciente_dependente é exemplo correspondente a *Categoria*, conforme mostrado na Figura 29.

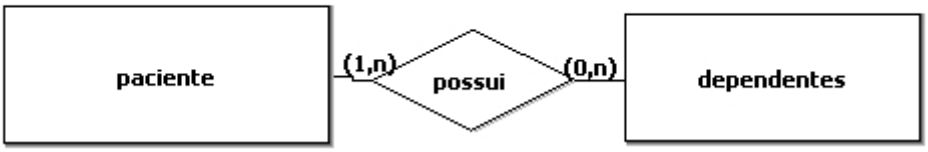

Figura 29: Exemplo de Categoria

4.1.7 Mapeamento da classificação de propriedade "papel formal"

A classificação de propriedade *Papel Formal* (GUARINO e WELTY, 2000a) não fornece CI (-O), não executa CI (-I), é antirrígido (~R) e dependente (+D), conforme visto na Tabela 1. Uma vez que não possui instâncias que possam ser identificadas, pois não executa identidade, conclui-se que o gerador de modelo de dados correspondente a tal elemento seria uma *entidade fraca*.

Com base nas restrições que as metapropriedades impõem aos relacionamentos hierárquicos, uma entidade correspondente a um *Papel Formal* pode ser uma entidade

de nível superior de entidades correspondentes a quaisquer tipos de propriedades não rígidas e dependentes. GUARINO e WELTY (2000a) recomendam que *Papéis Formais* subjuguem apenas *Papéis Materiais,* sendo a entidade que representa um *Papel Formal* utilizada para organizar a hierarquia de entidades correspondentes a papéis.

A entidade em questão pode ser entidade derivada somente de entidades correspondentes a *Categorias* e *Papéis Formais,* pois são os únicos tipos de propriedades que não executam CI.

Normalmente, assim como *Papel Material,* uma entidade correspondente a *Papel Formal* representa um papel desempenhado por uma entidade de nível superior em uma associação e representa os papéis mais genéricos do nível mais alto da hierarquia de papéis.

Exemplos de entidades correspondentes a *Papéis Formais* são: paciente e estudante. No caso de *Paciente,* como colocado em GUARINO e WELTY (2000a), a propriedade de ser *Paciente* de uma ação é um *Papel Formal*, pois não existe uma CI comum para pacientes em geral, uma vez que podem ser objetos, pessoas ou animais. A Figura 30 ilustra um exemplo de utilização de uma entidade a *Papel Formal*, como vinculada à entidade estudante, referente a *Papel Material*. Obviamente, como visto anteriormente, a entidade estudante também deverá ser vinculada a uma entidade correspondente a um *Tipo*, para herdar a CI que executa.

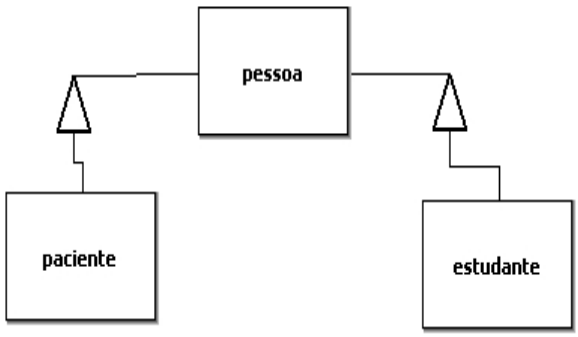

Figura 30: Exemplo a Papel Formal

Pode ser especializado ou generalizado de acordo com a modelagem conceitual e necessidades a serem atendidas por um sistema de informação específico.

4.1.8 Mapeamento da classificação de propriedade "atribuição"

A classificação de propriedade *Atribuição* (GUARINO e WELTY, 2000a), além de não fornecer e não executar CI (-O, -I), também é antirrígida e não dependente (~R, -D), ou semirrígida (¬R), conforme visto na Tabela 2.

Uma vez que não possui instâncias que possam ser identificadas e é antirrígido e não dependente, ou semirrígido, conclui-se que este tipo de propriedade corresponde a um *Atributo* de uma entidade.

A Tabela 3 mostra um resumo do que foi discutido acima, mostrando a associação das classificações de propriedades, obtidas a partir de combinações das metapropriedades vistas anteriormente, para a modelagem conceitual.

Tabela 3 Associação dos tipos de propriedades (GUARINO eWELTY 2000a)

Classificação da Propriedade (GUARINO e WELTY, 2000a)	**Meta-Propriedades**	**Construtor**
Tipo	+O +I +R	Ex.: Pessoa, Paciente
Quase-Tipo	-O +I +R	Ex.: Avaliação, Tipo_Avaliação
Papel Material	-O +I ~R +D	Ex.: Médico
Ordenável com Fase	-O +I ~R -D	Ex.: Tipos, Fases
Mixin	-O +I ¬R	Ex.: Prescrição médica ou medicamento
Categoria	-O -I +R	Ex.: Entidade Concreta, Entidade Abstrata
Papel Formal	-O -I ~R +D	Ex.: Paciente, Prescrição médica, medicamento,Instrumento

4.1.9 Passos aplicados na análise do modelo do hospital com o auxilio da ontologia:

- **Primeiro passo**: através da leitura dos metadados na busca e identificação das chaves primárias, chaves estrangeiras e tabelas auxiliares, bem como os relacionamentos (compostas apenas de chaves estrangeiras).

- **Segundo passo**: através dos relacionamentos (chaves estrangeiras) detectados, são feitas a leitura e identificação das tabelas/entidades que possuem vínculo.

- **Terceiro passo**: depois de identificadas as tabelas relacionadas, são realizadas varreduras nos atributos, verificando os tipos, através do *Data Type* e do *Nome* de cada um.

- **Quarto passo**: varredura é feita nos *Valores* contidos em cada atributo dessas tabelas, a fim de detectar equivalências por *Valores* inseridos.

- **Quinto passo**: Os atributos são categorizados em *Nominais* (textos/strings) e *numéricos* (valores em geral).

- **Sexto passo**: As tabelas sem utilização e as colunas detectadas com grande número de valores nulos ou brancos são excluídas da análise.

- **Sétimo passo**: Os atributos do tipo Data e Data/Hora têm seus valores separados em dia, mês e ano (dia, mês, ano, hora, minuto, segundo).

- **Oitavo passo**: Os atributos das tabelas analisadas são agrupados por equivalência de Data Type, Nome e Valores, respectivamente.

- **Nono passo**: Os atributos nominais são agrupados e ordenados; os numéricos são sumarizados, trazendo valor total por agrupamento, média, valor mínimo e máximo, além da contagem de atributos nominais cujo conjunto de valores é bastante limitado (sexo, estado civil e demais atributos).

- **Décimo passo**: É criado um metadado sintético/macro contendo os atributos equivalentes e disponibilizando os dados extraídos para análises de negócio.

Décimo primeiro passo: executar os algoritmos para determinar a possibilidade de o modelo gerado ser capaz de fornecer extração de conhecimento. Segundo (BORTOLETO e EBECKEN, 2008), estes passos constituem-se na adequação do modelo. Nas próximas seções, apresentam-se os modelos identificados para utilização na mineração de dados.

4.1.10 Discussão

Neste capítulo foi apresentada a metodologia para construção do modelo conceitual, em que se percebem os vínculos entre as entidades de acordo com as metapropriedades e a aplicação da CI para cada caso discutido. Esta compreensão nos permite construir o modelo de maneira adequada.

Os passos apresentados são proposições aplicadas a um modelo de Hospital mas podem ser estendidos de acordo com as características de cada modelo de dados a ser analisado, que é demonstrado no capítulo seguinte.

5 ESTUDO DE CASO

O trabalho foi executado com a utilização da base de dados do Hospital da Cruz Vermelha da cidade de Curitiba-PR, em que o modelo de dados foi dividido em três grandes áreas do protocolo hospitalar, que compreendem: a prescrição atendimento, prescrição médica e a prescrição solução. Cada área possui um conhecimento distinto sobre os dados que manipula.

O sistema que alimenta atualmente o banco de dados é o Tasy, da empresa Wheb Sistemas (TASY, 2010), que surgiu a partir da ideia de quatro médicos de Santa Catarina em criar um sistema capaz de gerenciar a tarefa hospitalar em seus diversos níveis, como farmácia, atendimento clínico, internamentos e outros. Por ser altamente adaptável, está implantado em diversos hospitais do Brasil. Na cidade de Curitiba, além do Cruz Vermelha, o Hospital Nossa Senhora das Graças e o Santa Cruz são hospitais que fazem uso do Sistema Tasy (TASY, 2010), que é utilizado como sistema de informação destes hospitais para gerenciar suas tarefas diárias. Foi desenvolvido utilizando a linguagem Borland Delphi e tem como gerenciador de banco de dados, no Cruz Vermelha, o Oracle 10g, com cerca de 2700 tabelas.

Sendo o sistema-mestre do Hospital da Cruz Vermelha, o sistema de informação Tasy é usado vastamente em todos os níveis em que ele se dispõe a atuar. Porém, sofre em alguns momentos pelo não preenchimento de dados - cerca de 60% das informações solicitadas em tela, até mesmo alguns campos-chave, que não são tratados contra nulos, geram exceções em algumas tarefas.

Apontada como a principal responsável por exceções, fechadas inesperadas e pelo mau uso, em algumas vezes, do sistema, está a necessidade da implantação rápida para sua utilização, aliada à falta de treinamento satisfatório.

O trabalho visa descrever alguns domínios, como prescrição atendimento, prescrição médica e prescrição solução, definindo os modelos conceituais para entendimento do sistema através da metodologia proposta e validando os modelos para utilização posterior de forma adequada, visto que a documentação é precária e a há a não utilização de consultas pelo desconhecimento do modelo e pela complexidade do mesmo.

O trabalho foi feito em três etapas: análise do modelo de dados, criação dos domínios por área de negócio com dados consolidados e a criação de um metamodelo de dados que conheçae as informações granulares através da ontologia.

Todas as validações foram feitas com os profissionais da área de TI do hospital e com supervisão do Dr. Angelmar Roman para esclarecimento das dúvidas e construção do modelo conceitual.

A primeira fase compreendeu a análise do modelo que efetivamente era usado pelo sistema de informação Tasy, a fim de se definir se os domínios trariam informações para a criação dos modelos conceituais.

Na segunda fase, definiu-se a construção dos domínios através do uso das ontologias, validando os modelos

Na terceira fase, foram realizados testes com mineração para verificar a possibilidade de uso do modelo, de acordo coma metodologia de criação dos domínios.

5.1 Prescrição atendimento

Refere-se às informações a respeito do paciente dentro do hospital, como laudos, movimento no estoque (medicação utilizada, por exemplo), sua nutrição e demais informações relacionadas ao atendimento dispensado ao paciente. Possui classificação segundo o tipo de atendimento dispensado. Modelo importante para se mapear a estadia do cliente segundo alguns aspectos, como seu tratamento em função de idade e sexo. Nesses parâmetros se define quais as incidências de doenças em uma determinada época do ano. É também capaz de definir dosagem de medicamentos e materiais usados em tratamentos de certas doenças segundo alguns parâmetros pessoais do paciente, como peso, por exemplo, dando a oportunidade de um planejamento nos custos do atendimento.

Para a construção do diagrama entidade-relacionamento, foram aplicados o primeiro e o segundo passo, mostrados na figura 31.

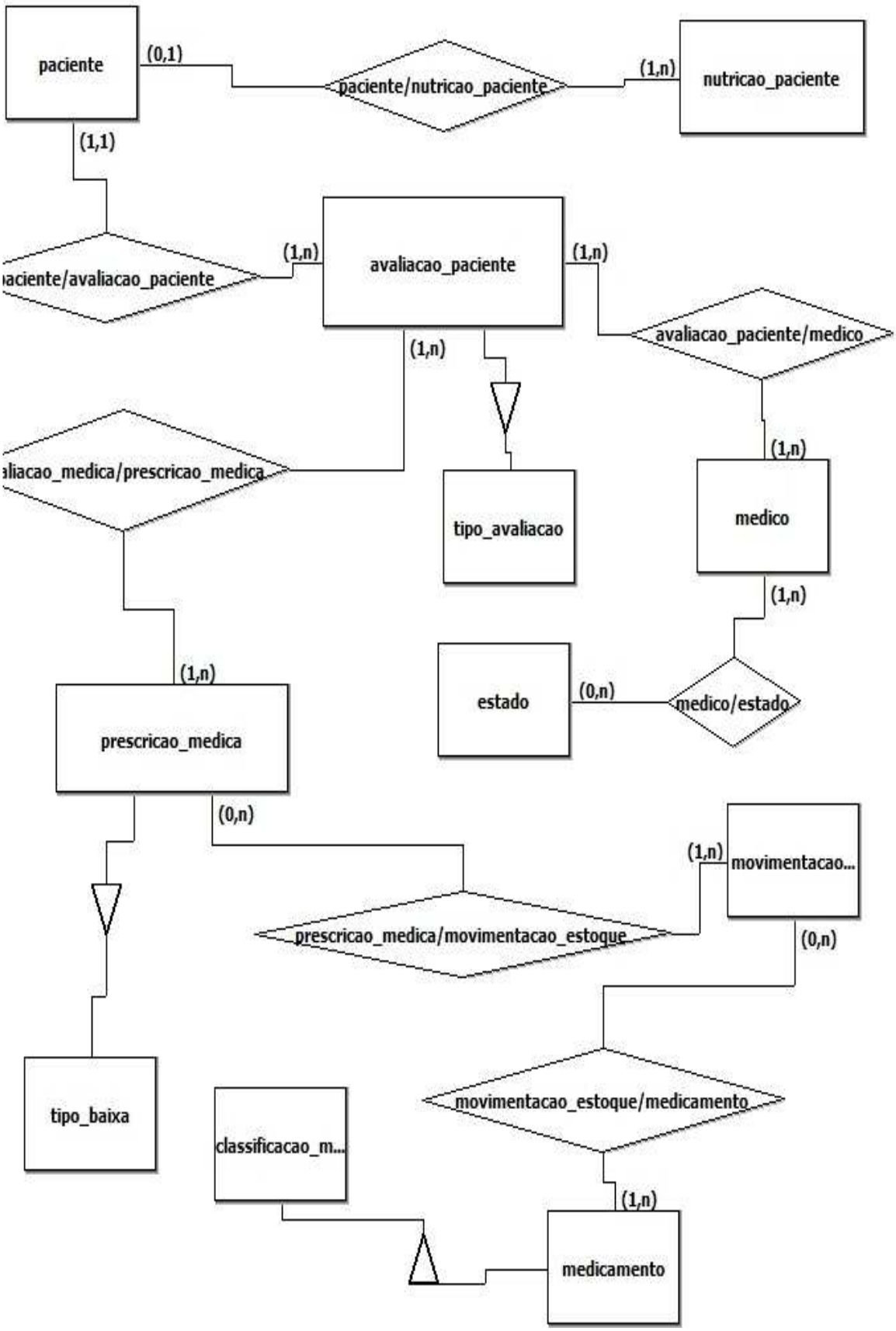

Figura 31: DER, prescrição atendimento

Para a construção do modelo entidade-relacionamento, após a construção do diagrama entidade-relacionamento, foram inseridos os atributos e excluídas as tabelas que não contêm dados e ou sem utilização pelo sexto passo

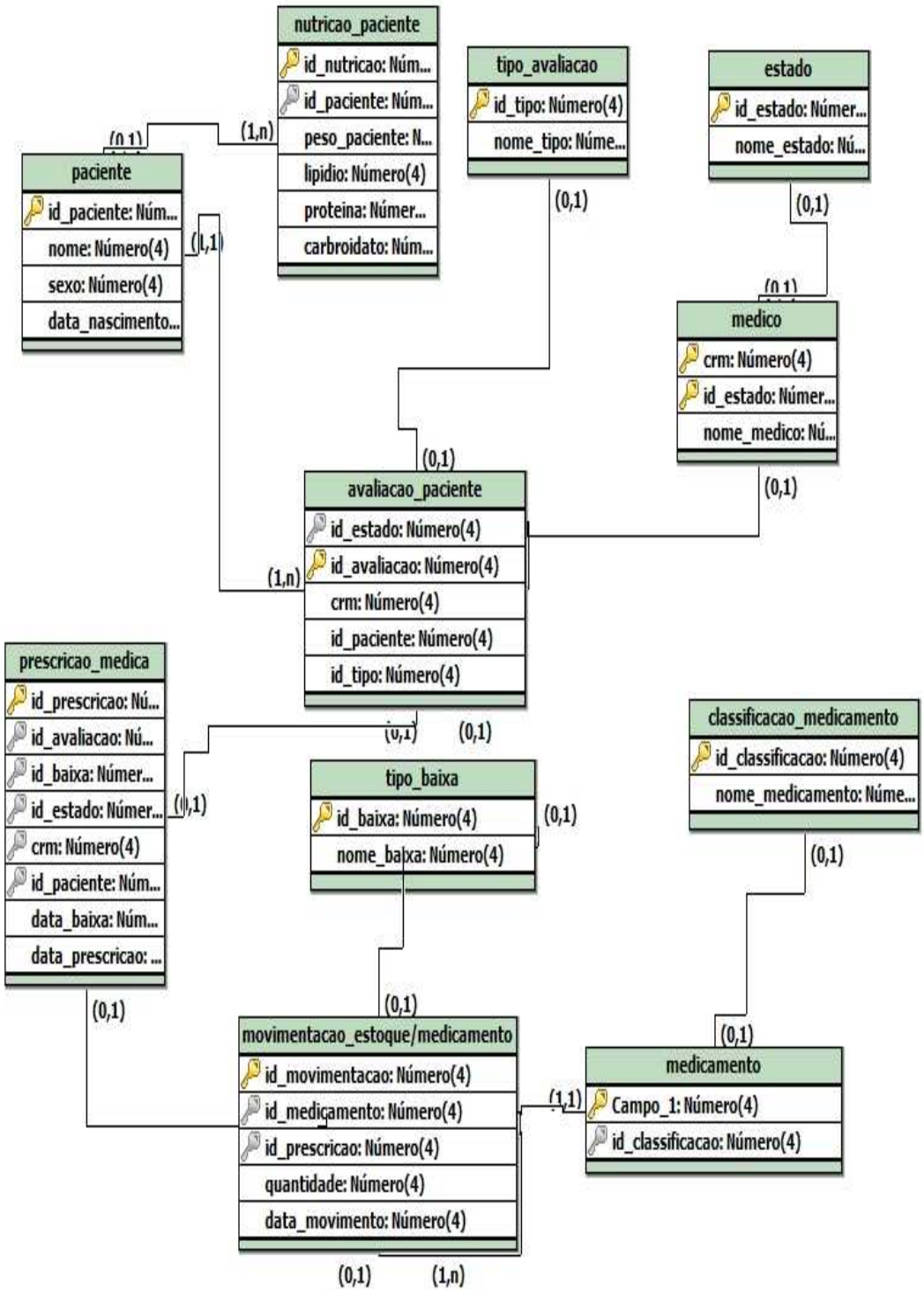

Figura 32: Modelo lógico: prescrição atendimento

A tabela 4 representa as entidades com suas respectivas chaves primárias e atributos correspondentes e se estas são algum subtipo ou são principais, denominadas de raiz, após aplicação do primeiro passo.

Tabela 4: Representação das entidades: prescrição atendimento

Entidade	**Chave Primária**	**Atributos**
Paciente	id_paciente	nome_paciente, sexo data_nascimento
nutricao_paciente	id_nutricao	id_paciente, peso_paciente Lipídio,proteína carboidrato
Estado	id_estado	Nome_estado
Medico	crm,id_estado	nome_medico
tipo_avaliacao	id_tipo	nome_tipo
avaliacao_paciente	Id_avaliacao	id_estado,crm id_paciente id_tipo
tipo_baixa	id_baixa	nome_baixa
prescricao_medica	Id_prescricao	id_avaliacao,id_baixa id_estado,CRM, id_paciente, data_prescricao, data_baixa
classificacao_ medicamento	id_classificacao	nome_classificacao
medicamento	id_medicamento	id_classificacao, nome_medicamento
movimentacao_estoque	id_movimento	id_medicamento, id_prescricao, Quantidade,valor, data_movimento

A Tabela 5 representa as entidades e seus relacionamentos, bem como a dependência identificada como imagem e as cardinalidades entre as entidades, obtidas pelo segundo passo.

Tabela 5: Representação dos relacionamentos*: prescrição atendimento*

Relações	Domínio	Imagem	Cardinalidade
			Domínio
paciente/avaliação	paciente	avaliação	(1,1)
avaliação/paciente	avaliação	paciente	(1,N)
paciente/nutricao	paciente	nutrição	(0,N)
nutrição/paciente	nutrição	paciente	(1,N)
Avaliação/medico	avaliação	medico	(1,N)
medico/avaliação	medico	avaliação	(1,N)
medico/estado	medico	estado	(1,1)
estado/medico	estado	medico	(0,N)
avaliação/prescricao	avaliação	prescrição	(1,N)
prescrição /avaliação	prescrição	avaliação	(1,1)
prescrição /movimento[4]	prescrição	movimento	(0,N)
movimento/prescrição	movimento	prescrição	(1,N)
movimento/medicamento	movimento	medicamento	(0,N)
medicamento/movimento	medicamento	movimento	(1,N)

A Tabela 6 representa o conceito dos metadados com descrição e entidades respectivamente, obtidas pelo quarto passo.

Tabela 6: Documentações dos conceitos*: prescrição atendimento (parte 1)*

Nome do conceito	PACIENTE
Descrição do conceito	É todo aquele que gera ação para iniciar as demais operações, seja por consulta, exame, internamento ou cirurgia (ou alguma combinação possível entre esses eventos).
Sinônimo	Cliente.
Nome do conceito	NUTRICAO_PACIENTE
Descrição do conceito	Refere-se aos hábitos de alimentação recomendados a um paciente.
Sinônimo	-
Nome do conceito	ESTADO
Descrição do conceito	Local de origem do CRM do médico.
Sinônimo	-
Nome do conceito	MEDICO
Descrição do conceito	É um componente responsável pela avaliação e pela prescrição de medicamentos de um paciente.
Sinônimo	Profissional de saúde (não confundir com enfermeiros).
Nome do conceito	TIPO_AVALIACAO

Tabela 7: Documentações dos Conceitos*: prescrição atendimento (parte 2)*

Descrição do conceito	Refere-se aos tipos que uma avaliação médica (AVALIACAO_PACIENTE) pode assumir e irá direcionar o restante do tratamento.
Sinônimo	-
Nome do conceito	AVALIACAO_PACIENTE
Descrição do conceito	É uma especialização do tipo de avaliação. Identifica quem fez ou quem recebeu esse tipo de avaliação, além da data em que ela foi feita.
Sinônimo	Diagnóstico.
Nome do conceito	TIPO_BAIXA
Descrição do conceito	São estados em que o paciente pode se encontrar ao final de uma análise médica (cura, óbito, internamento).
Sinônimo	Tipo de alta.
Nome do conceito	PRESCRICAO_MEDICA
Descrição do conceito	É uma especialização do tipo de baixa. Nela estão contidos quem executou e quem foi analisado, qual a conclusão e quando foi feita essa análise.
Sinônimo	-
Nome do conceito	CLASSIFICACAO_MEDICAMENTO
Descrição do conceito	São formas de classificação que um determinado medicamento pode assumir segundo sua fórmula (genérico, similar ou referência).
Sinônimo	-
Nome do conceito	MEDICAMENTO
Descrição do conceito	É a solução química ministrada ao paciente, com o intuito de curá-lo.
Sinônimo	Remédio.
Nome do conceito	MOVIMENTACAO_ESTOQUE
Descrição do conceito	São movimentos de saída de medicamentos do estoque para uso em pacientes.
Sinônimo	-

A tabela 7 representa as propriedades, com nome do atributo, tipo de dados (imagem), entidade a que pertence denominado domínio, tipo que é definido por datatype (tipo de dados) ou object (associação entre entidades) e pela característica que denota se é um atributo funcional (necessário) ou não, obtido através do terceiro passo, quinto passo, sexto passo,sétimo passo e oitavo passo

Tabela 8: Atributos transformados em propriedades: *datatype e object (prescriçãoatendimento) (parte 1)*

Nome	Tipo	Domínio	Imagem	Característica
Id_paciente	Datatype	PACIENTE	Inteiro	Funcional
Nome_paciente	Datatype	PACIENTE	Texto	Funcional
Sexo	Datatype	PACIENTE	Lógico	Funcional
Data_nascimento	Datatype	PACIENTE	Data	Funcional
Paciente/Nutricao_paciente	Object	PACIENTE	-	Funcional
Paciente/ Avaliacao_paciente	Object	PACIENTE	-	Funcional
Id_paciente	Datatype	NUTRICAO_PACIENTE	Inteiro	Funcional
Peso_paciente	Datatype	NUTRICAO_PACIENTE	Decimal	Funcional
Lipidio	Datatype	NUTRICAO_PACIENTE	Decimal	Funcional
Proteina	Datatype	NUTRICAO_PACIENTE	Decimal	Funcional
Carboidrato	Datatype	NUTRICAO_PACIENTE	Decimal	Funcional
Id_nutricao	Datatype	NUTRICAO_PACIENTE	Inteiro	Funcional
Id_estado	Datatype	ESTADO	Inteiro	Funcional
Nome_estado	Datatype	ESTADO	Texto	Funcional
Medico/Estado	Object	ESTADO	-	Funcional
Crm	Datatype	MEDICO	Inteiro	Funcional
Id_estado	Datatype	MEDICO	Inteiro	Funcional
Nome_medico	Datatype	MEDICO	Texto	Funcional
Avaliacao_paciente/ Medico	Object	MEDICO	-	Funcional
Id_tipo	Datatype	TIPO_AVALIACAO	Inteiro	Funcional
Nome_tipo	Datatype	TIPO_AVALIACAO	Texto	Funcional
Id_avaliacao	Datatype	AVALIACAO_PACIENTE	Inteiro	Funcional
Id_estado	Datatype	AVALIACAO_PACIENTE	Inteiro	Funcional
Crm	Datatype	AVALIACAO_PACIENTE	Inteiro	Funcional
Id_paciente	Datatype	AVALIACAO_PACIENTE	Inteiro	Funcional
Id_tipo	Datatype	AVALIACAO_PACIENTE	Inteiro	Funcional
Avaliacao_paciente/ Prescricao_medica	Object	AVALIACAO_PACIENTE	-	Funcional
Id_baixa	Datatype	TIPO_BAIXA	Inteiro	Funcional
Nome_baixa	Datatype	TIPO_BAIXA	Texto	Funcional
Id_prescricao	Datatype	PRESCRICAO_MEDICA	Inteiro	Funcional
Id_avaliacao	Datatype	PRESCRICAO_MEDICA	Inteiro	Funcional
Id_baixa	Datatype	PRESCRICAO_MEDICA	Inteiro	Funcional
Id_estado	Datatype	PRESCRICAO_MEDICA	Inteiro	Funcional
Crm	Datatype	PRESCRICAO_MEDICA	Inteiro	Funcional
Id_paciente	Datatype	PRESCRICAO_MEDICA	Inteiro	Funcional

Tabela 9: Atributos transformados em propriedades: *datatype e objeto (prescrição atendimento) (parte 2)*

Nome	Tipo	Domínio	Imagem	Característica
Data_prescricao	Datatype	PRESCRICAO_MEDICA	Data	Funcional
Data_baixa	Datatype	PRESCRICAO_MEDICA	Data	Funcional
Prescricao_medica/ Movimentacao_estoque	Object	PRESCRICAO_MEDICA	-	Funcional
Id_classificacao	Datatype	CLASSIFICACAO_MEDICAMENTO	Inteiro	Funcional
Nome_classificacao	Datatype	CLASSIFICACAO_MEDICAMENTO	Texto	Funcional
Id_medicamento	Datatype	MEDICAMENTO	Inteiro	Funcional
Id_classificacao	Datatype	MEDICAMENTO	Inteiro	Funcional
Nome_medicamento	Datatype	MEDICAMENTO	Texto	Funcional
Id_movimentacao	Datatype	MOVIMENTACAO_ESTOQUE	Inteiro	Funcional
Id_medicamento	Datatype	MOVIMENTACAO_ESTOQUE	Inteiro	Funcional
Id_prescricao	Datatype	MOVIMENTACAO_ESTOQUE	Inteiro	Funcional
Quantidade	Datatype	MOVIMENTACAO_ESTOQUE	Decimal	Funcional
Valor	Datatype	MOVIMENTACAO_ESTOQUE	Decimal	Funcional
Data_movimento	Datatype	MOVIMENTACAO_ESTOQUE	Data	Funcional
Movimentacao_estoque/ Medicamento	Object	MOVIMENTACAO_ESTOQUE	-	Funcional

A tabela 8 representa as restrições entre as entidades, demonstrando o nome do conceito, tipo de restrição (se deve ter algum valor), tipo de condição (se é necessário, suficiente ou ambas) e notação, que denota a relação entre as entidades, que representam o nono passo.

Tabela 10: Representação das restrições: *prescrição atendimento (parte 1)*

Nome do Conceito	Tipo da Restrição	Tipo da Condição	Notação
PACIENTE	SOME VALUES OF	Necessária	PACIENTE: ? paciente/nutricao_paciente. nutricao_paciente
NUTRICAO_PACIENTE	ALL VALUES OF	Necessária e Suficiente	PACIENTE: ? paciente/nutricao_paciente. Paciente
PACIENTE	SOME VALUES OF	Necessária	PACIENTE: ??paciente/avaliacao_ paciente.avaliacao_ paciente

Tabela 11: Representação das restrições: *prescrição atendimento (parte 2)*

Nome do Conceito	Tipo da Restrição	Tipo da Condição	Notação
AVALIACAO_PACIENTE	INTER-SECTION OF	Necessária e Suficiente	AVALIACAO_ PACIENTE: avaliacao_paciente/ paciente.paciente ?AVALIACAO_ PACIENTE: avaliacao_paciente/ paciente.paciente ? 1
AVALIACAO_PACIENTE	SOME VALUES OF	Necessária	AVALIACAO_ PACIENTE: avaliacao_paciente/ medico.medico
MEDICO	SOME VALUES OF	Necessária	MEDICO: avaliacao_ paciente/medico. avaliacao_paciente
MEDICO	SOME VALUES OF	Necessária	MEDICO: medico/estado.medico
ESTADO	ALL VALUES FROM	Necessária e Suficiente	ESTADO: medico/estado.medico
PRESCRICAO_MEDICA	SOME VALUES OF	Necessária	PRESCRICAO_MEDICA: avaliacao_paciente/ prescricao_medica. avaliacao_paciente
AVALIACAO_PACIENTE	INTER-SECTION OF	Necessária e Suficiente	AVALIACAO_ PACIENTE: avaliacao_paciente/ prescricao_medica.prescricao_ medicaAVALIACAO_ PACIENTE: avaliacao_paciente/ prescricao_medica.prescricao_ medica
PRESCRICAO_MEDICA	SOME VALUES OF	Necessária	PRESCRICAO_MEDICA: movimentacao_estoque/ prescricao_medica. movimentacao_estoque
MOVIMENTACAO_ESTOQUE	ALL VALUES FROM	Necessária	MOVIMENTACAO_ ESTOQUE: precricao_medica/ movimentacao_estoque. prescrição_medica
MOVIMENTACAO_ESTOQUE	SOME VALUES OF	Necessária	MOVIMENTACAO_ ESTOQUE: movimentacao_estoque/ medicamento.medicamento
MEDICAMENTO	ALL VALUES FROM	Necessária e Suficiente	MEDICAMENTO: movimentacao_ estoque/medicamento. movimentacao_estoque

5.1.1 Atribuição das metapropriedades às entidades do domínio prescrição Atendimento

5.1.1.1 Paciente

A entidade Paciente armazena informações pessoais sobre os pacientes do Hospital. A seguir, é realizada uma análise da entidade *Cliente*, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (~R) – todo Paciente não será necessariamente um Cliente durante toda a sua existência.
- Não fornece CI (-O) – desde que uma mesma pessoa possa ser um Paciente diferente em diferentes prescrições de atendimento, uma condição de identidade (CI) fornecida por Paciente pode ser apenas local, dentro de uma determinada situação de atendimento.
- Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.
- Dependente (+D) – o paciente será considerado externamente dependente de avaliação, pois, diferente do conceito de Paciente, fornecido por MONTEIRO (2002), alguém somente poderá ser um Paciente de uma determinada avaliação se houver recebido alguma prescrição de atendimento da mesma.

Após esta análise, conclui-se que a entidade Paciente pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.1.2 Nutrição_Paciente

A entidade Nutrição_Paciente refere-se aos hábitos de alimentação recomendados a um paciente.

A seguir, é realizada uma análise da entidade Nutrição_Paciente, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirrígida (~R) – A Nutrição_Paciente não será necessariamente aplicada a paciente durante toda a sua existência.

• Não fornece CI (-O) – desde que Nutrição_Paciente possa ser aplicado a um paciente diferentes vezes em diferentes prescrições de atendimento, uma condição de identidade (CI) fornecida por Paciente pode ser apenas local, dentro de uma determinada situação de prescrição.

• Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.

• Dependente (+D) – Paciente será considerado externamente dependente de avaliação, pois, diferente do conceito de Paciente, fornecido por MONTEIRO (2002), alguém somente poderá ser um Paciente de uma determinada avaliação se houver recebido alguma prescrição de atendimento da mesma.

Após esta análise, conclui-se que a entidade Nutrição_Paciente pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.1.3 Estado

A entidade Estado armazena informações sobre o local de origem do CRM do médico. A seguir, é realizada uma análise da entidade Estado, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirrígida (+R) – Estado sempre será vinculado ao CRM durante toda a sua existência.

• Não Fornece CI (-O) – o Estado pode ser identificado globalmente por meio de uma característica própria. Em diferentes locais, uma condição de identidade (CI) é fornecida , dentro de uma determinada situação de atendimento.

• Não Executa CI (-I) – executa uma CI relacionada à pessoa, de um modo geral.

• Dependente (+D) – é externamente dependente de médico, pois um médico somente irá existir se houver um Estado ao qual é referente. Fornecido por MONTEIRO (2002), alguém somente poderá ser um médico de uma determinada avaliação, se houver Estado.

Após esta análise, conclui-se que a entidade Estado pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Categoria*.

5.1.1.4 Médico

A entidade Médico armazena informações e é um componente responsável pela avaliação e pela prescrição de medicamentos de um paciente

A seguir, é realizada uma análise da entidade Médico, com relação a cada uma das metapropriedades vistas anteriormente:

•Antirrígida (~R) – todo médico existirá necessariamente em uma prescrição_atendimento durante toda a sua existência.
•Não fornece CI (-O) – desde que uma mesma pessoa pode ser um médico diferentes vezes em diferentes prescrições de atendimento, uma condição de identidade (CI) fornecida por Médico pode ser apenas local, dentro de uma determinada situação de atendimento.
•Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.
•Dependente (+D) – Médico será considerado externamente dependente de avaliação. Fornecido por MONTEIRO (2002), alguém somente poderá ser um médico de uma determinada avaliação se houver prescrito um atendimento.

Após esta análise, conclui-se que a entidade Médico pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.1.5 Tipo_Avaliação

A entidade Tipo_Avaliação armazena informações referentes aos tipos que uma avaliação médica (AVALIACAO_PACIENTE) pode assumir e irá direcionar o restante do tratamento

A seguir, é realizada uma análise da entidade Tipo_Avaliação, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirrígida (+R) – toda instância de Tipo_Avaliação será necessariamente durante toda a sua existência

• Não fornece CI (+O) – todo Tipo_Avaliação pode ser identificado globalmente (ou seja, durante toda a sua existência) por meio de uma característica própria. •

Executa CI (+I) – uma vez que fornece uma condição de identidade, obviamente também a executa.

• Dependente (+D) – é externamente dependente de Avaliação_Paciente, pois um Tipo_Avaliação existirá, se houver um Avaliação_Paciente ao qual é referente.

Após esta análise, conclui-se que a entidade Tipo_Avaliação pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Tipo*.

5.1.1.6 Avaliação_Paciente

A entidade Avaliação_Paciente *é* uma especialização do tipo de avaliação. Armazena informações sobre quem fez e quem recebeu esse tipo de avaliação, além da data em que ela foi feita.

A seguir, é realizada uma análise da entidade Avaliação_Paciente, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirrígida (+R) – toda Avaliação_Paciente será necessariamente vinculada a um paciente durante toda a sua existência.

• Não fornece CI (-O) – desde que uma mesma pessoa possa ser um Paciente diferentes vezes em diferentes prescrições de atendimento, uma condição de identidade (CI) é fornecida por Avaliação_Paciente, dentro de uma determinada situação de atendimento.

• Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.

• Dependente (+D) – Avaliação_Paciente será considerado externamente dependente de Paciente. Fornecido por MONTEIRO (2002), alguém somente poderá ser um Paciente de uma determinada avaliação se houver recebido alguma prescrição de atendimento da mesma.

Após esta análise, conclui-se que a entidade Avaliação_Paciente pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.1.7 Tipo_Baixa

A entidade Tipo_Baixa armazena informações sobre os estados em que o paciente pode se encontrar ao final de uma análise médica (cura, óbito, internamento).

A seguir, é realizada uma análise da entidade Tipo_Baixa, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – Tipo_Baixa existirá durante toda a sua existência.
- Não fornece CI (-O) – nenhum Tipo_Baixa pode ser identificado globalmente por meio de uma característica própria.
- Não executa CI (-I) – executa uma CI relacionada à Prescrição_Medica de um modo geral.
- Dependente (+D) – é externamente dependente de Prescrição_Medica e somente irá existir se houver uma Prescrição_Medica e o estado ao qual é referente, fornecido por MONTEIRO (2002),

Após esta análise, conclui-se que a entidade Tipo_Baixa pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um Tipo.

5.1.1.8 Prescrição_Medica

A entidade Prescrição_Medica armazena informações sobre quem executou e quem foi analisado, qual a conclusão e quando foi feita essa análise.

A seguir, é realizada uma análise da entidade Prescrição_Medica, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – toda Prescrição_Medica existirá se houver Avaliação_Paciente.
- Não fornece CI (-O) – desde que uma mesma pessoa possa ser um Paciente diferentes vezes em diferentes prescrições de atendimento, uma condição de identidade (CI) é fornecida por Prescrição_Medica, dentro de uma determinada situação de atendimento.
- Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.
- Dependente (+D) – Prescrição_Medica será considerado externamente dependente de Avaliação_Paciente, pois, fornecido por MONTEIRO (2002), alguém somente poderá ter um Prescrição_Medica de uma determinada avaliação.

Após esta análise, conclui-se que a entidade Prescrição_Medica pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.1.9 Classificação_Medicamento

A entidade Classificação_Medicamento armazena informações sobre a classificação que um determinado medicamento pode assumir segundo sua fórmula (genérico, similar ou referência)

A seguir, é realizada uma análise da entidade Classificação_Medicamento, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – Classificação_Medicamento terá vinculo com medicamento durante toda a sua existência.
- Não fornece CI (-O) – nenhuma Classificação_Medicamento pode ser identificada globalmente por meio de uma característica própria. Em diferentes locais, uma condição de identidade (CI) fornecida por Classificação_Medicamento pode ser apenas local, dentro de um determinado medicamento.
- Não executa CI (-I) – executa uma CI relacionada a medicamento, de um modo geral.
- Dependente (+D) – é externamente dependente de medicamento, pois o mesmo só irá existir,se houver um Classificação_Medicamento ao qual é referente. Fornecido por MONTEIRO (2002), medicamento só existirá se houver Classificação_Medicamento.

Após esta análise, conclui-se que a entidade Classificação_Medicamento pode ser classificada, de acordo com GUARINO e WELTY (2000a), como uma Categoria.

5.1.1.10 Medicamento

A entidade Medicamento armazena informações sobre a solução química ministrada ao paciente, com o intuito de curá-lo

A seguir, é realizada uma análise da entidade Medicamento, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – medicamento existirá durante toda a sua existência.
- Não fornece CI (-O) – nenhum medicamento pode ser identificado globalmente por meio de uma característica própria.
- Não executa CI (-I) – executa uma CI relacionada à

Classificação_Medicamento, de um modo geral.

- Dependente (+D) – é externamente dependente de Classificação_Medicamento e somente irá existir se houver uma classificação ao qual é referente, fornecido por MONTEIRO (2002),

Após esta análise, conclui-se que a entidade Medicamento pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um Tipo.

5.1.1.11 Movimentação_Estoque

A entidade Movimentação_Estoque armazena informações sobre a saída de medicamentos do estoque para uso em pacientes

A seguir, é realizada uma análise da entidade Movimentação_Estoque, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – Movimentação_Estoque existirá durante toda a sua existência.
- Não fornece CI (+O) – Movimentação_Estoque não pode ser identificado globalmente por meio de uma característica própria.
- Não executa CI (-I) – executa uma CI relacionada a Prescrição_Medica, de um modo geral.
- Dependente (+D) – é externamente dependente de Prescrição_Medica e irá existir se houver uma Prescrição_Medica ao qual é referente, fornecido por MONTEIRO (2002),

Após esta análise, conclui-se que a entidade Movimentação_Estoque pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um Tipo

5.1.1.12 Prescrição médica

Quando o médico prescreve um determinado medicamento a um paciente, algumas vezes o medicamento indicado não está sendo destinado a solucionar em definitivo o problema do paciente, mas sim prepará-lo para algum outro tratamento. Também existe as questões cirúrgicas, que são procedimentos executados dentro do hospital e são destinados à terapia do pacientes.

Para a construção do diagrama entidade-relacionamento, foram aplicados o primeiro e o segundo passo, mostrados na figura 33.

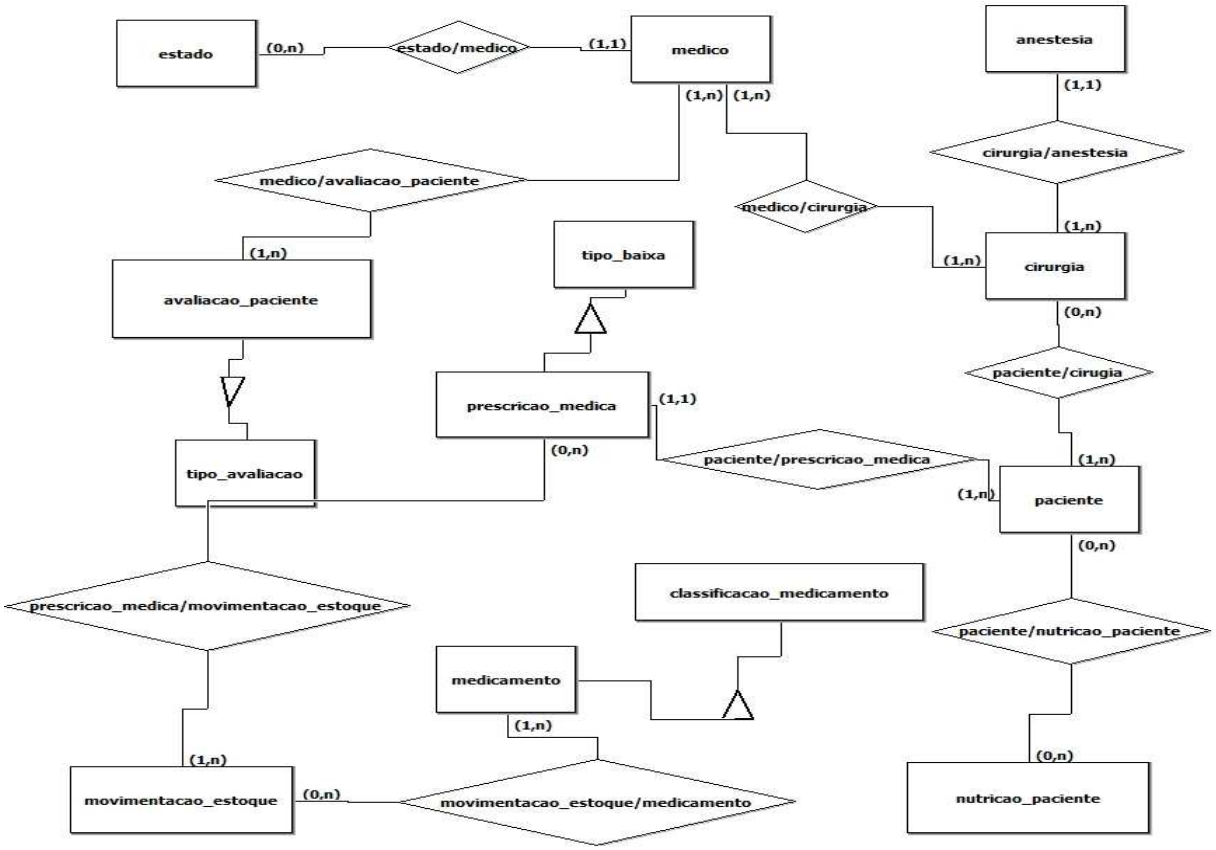

Figura 33: DER: prescrição médica

Para a construção do modelo entidade-relacionamento, após a construção do diagrama entidade-relacionamento, foram inseridos os atributos e excluídas as tabelas que não contêm dado e não têm utilização pelo sexto passo

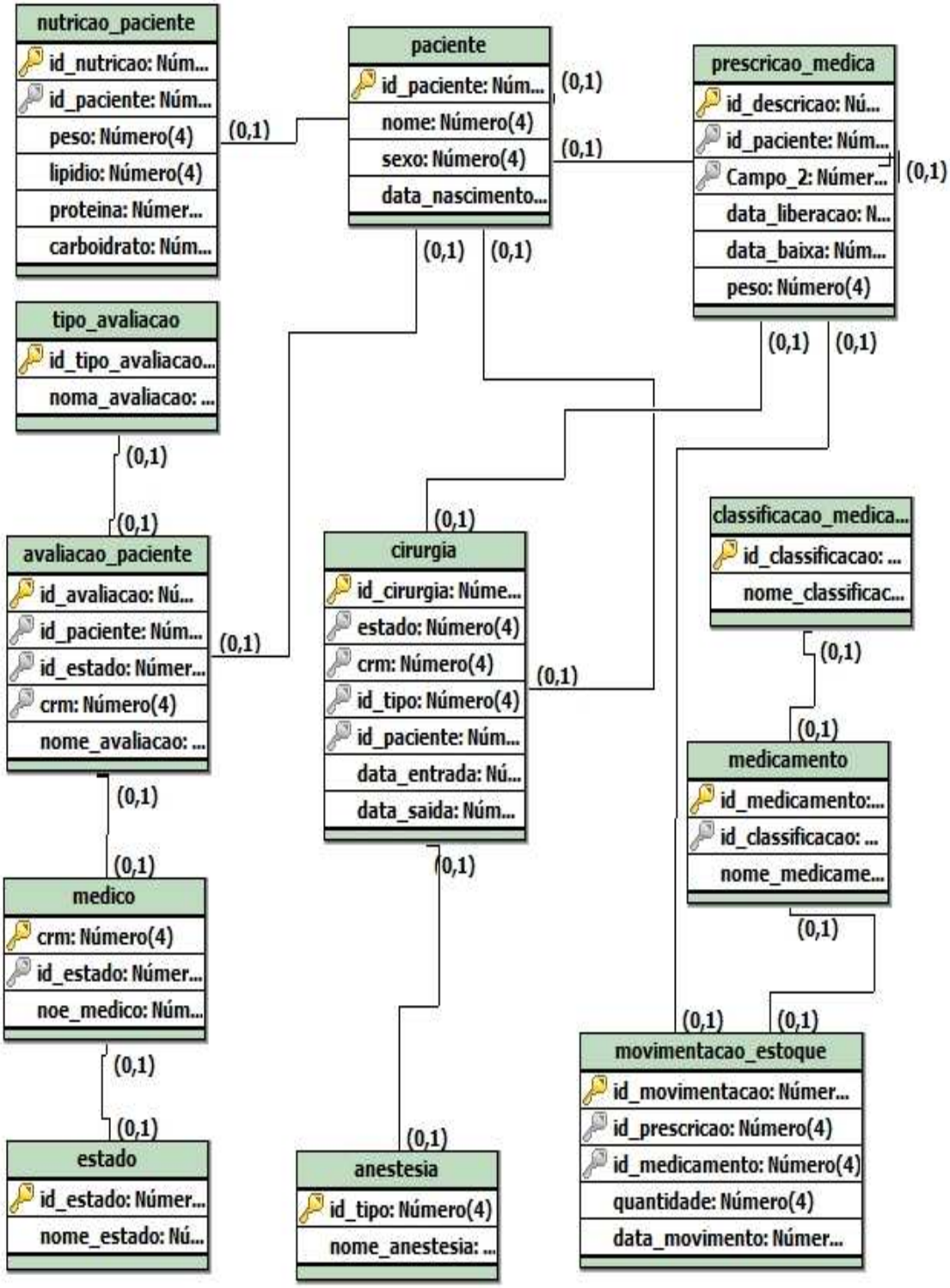

Figura 34:Modelo Lógico: Prescrição Médica

A Tabela 9 representa as entidades, com suas respectivas chaves primárias e atributos correspondentes, e se estas são algum subtipo ou são principais, denominadas de raiz, após a aplicação do primeiro passo.

Tabela 12: Representação de entidades: *prescrição médica*

Entidade	**Chave Primária**	**Atributos**
Cirurgia	id_cirurgia	id_tipo id_estado id_paciente crm data_entrada data_saida
Anestesia	id_tipo	nome_anestesia

A Tabela 10 representa as entidades e seus relacionamentos, bem como a dependência identificada como imagem, as cardinalidades e as relações inversas entre as entidades, obtidas pelo segundo passo.

Tabela 13:. Representação dos relacionamentos: *prescrição médica*

Relações	**Domínio**	**Imagem**	**Cardinalidade**	
			Domínio	**Imagem**
medico/cirurgia	medico	cirurgia	(1,N)	(1,N)
cirurgia/medico	cirurgia	medico	(1,N)	(1,N)
anestesia/cirurgia	anestesia	cirurgia	(1,1)	(1,N)
cirurgia/anestesia	cirurgia	anestesia	(1,N)	(1,1)
paciente/cirurgia	paciente	cirurgia	(1,N)	(0,N)
cirurgia/paciente	cirurgia	paciente	(0,N)	(1,N)

A Tabela 11 representa o conceito dos metadados, com descrição e entidades respectivamente, obtidas pelo quarto passo.

Tabela 14: Documentações dos conceitos: *prescrição médica*

Nome do conceito	CIRURGIA
Descrição do conceito	São tratamentos executados nos pacientes visando sua cura. Não estão inseridas cirurgias para exame, como, por exemplo, biópsias.
Sinônimo	Intervenção cirúrgica.
Nome do conceito	ANESTESIA
Descrição do conceito	Não é uma especialização de medicamento por seu efeito não tratar da cura. Apenas prepara para uma intervenção cirúrgica ou aplicação que gere dor.
Sinônimo	-

A Tabela 12 representa as propriedades, com nome do atributo, tipo de dados (imagem), entidade a que pertence denominado domínio, tipo que é definido por datatype (tipo de dados) ou object (associação entre entidades) e pela característica que denota se é um atributo funcional (necessário) ou não, obtidas através do terceiro passo, quinto passo e sétimo passo.

Tabela 15: Atributos transformados em propriedades: *datatype e object (prescrição médica)*

Nome	**Tipo**	**Domínio**	**Imagem**	**Característica**
Id_cirurgia	Datatype	CIRURGIA	Inteiro	Funcional
Id_tipo	Datatype	CIRURGIA	Inteiro	Funcional
Id_estado	Datatype	CIRURGIA	Inteiro	Funcional
Id_paciente	Datatype	CIRURGIA	Inteiro	Funcional
Crm	Datatype	CIRURGIA	Inteiro	Funcional
Data_entrada	Datatype	CIRURGIA	Data	Funcional
Data_saida	Datatype	CIRURGIA	Data	Funcional
Cirurgia/Anestesia	Object	CIRURGIA	-	Funcional
Id_tipo	Datatype	ANESTESIA	Inteiro	Funcional
Nome_anestesia	Datatype	ANESTESIA	Texto	Funcional

A Tabela 13representa as restrições entre as entidades, demonstrando o nome do conceito, tipo de restrição (se deve ter algum valor), tipo de condição (se é necessário, suficientes ou ambos) e notação, que denota a relação entre as entidades, que representam o oitavo e novo passos**.**

Tabela 16: Representação das restrições: *prescrição médica*

Nome do Conceito	**Tipo da Restrição**	**Tipo da Condição**	**Notação em DL**
MEDICO	SOME VALUES OF	Necessária	MEDICO: medico/cirurgia.cirurgia
CIRURGIA	SOME VALUES OF	Necessária	CIRURGIA: medico/cirurgia.medico
MEDICO	SOME VALUES OF	Necessária	MEDICO: medico/estado.medico
CIRURGIA	ALL VALUES FROM	Necessária e suficiente	CIRURGIA: cirurgia/anestesia.anestesia
ANESTESIA	SOME VALUES OF	Necessária	ANESTESIA: cirurgia/anestesia. cirurgia

5.1.2 Atribuição das metapropriedades às entidades do domínio prescrição médica

5.1.2.1 Cirurgia

A entidade Cirurgia armazena informações sobre tratamentos executados nos pacientes visando sua cura. Não estão inseridas cirurgias para exame, como, por exemplo, biópsias pessoais nos pacientes do hospital.

A seguir, é realizada uma análise da entidade Cirurgia, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – toda cirurgia será necessariamente feita em um paciente.
- Não fornece CI (-O) – desde que uma mesma pessoa pode sofrer uma cirurgia diferentes vezes, uma condição de identidade (CI) fornecida por cirurgia pode ser apenas local.
- Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.
- Dependente (+D) – Cirurgia será considerada externamente dependente de paciente e médico, pois, diferente do conceito de Paciente, fornecido por MONTEIRO (2002), alguém somente poderá sofrer uma cirurgia de uma determinada avaliação se houver recebido alguma prescrição médica da mesma.

Após esta análise, conclui-se que a entidade Cirurgia pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.2.2 Anestesia

A entidade Anestesia armazena informações sobre o medicamento por seu efeito para uma intervenção cirúrgica ou aplicação que gere dor.

A seguir, é realizada uma análise da entidade Anestesia, com relação a cada uma das metapropriedades vistas anteriormente:

. • Antirrígida (+R) – anestesia existirá durante toda a sua existência.

. • Não fornece CI (-O) – nenhuma anestesia pode ser identificada globalmente por meio de uma característica própria.

. • Não executa CI (-I) – executa uma CI relacionada à cirurgia, de um modo geral.

. • Dependente (+D) – é externamente vinculada à cirurgia e somente irá existi, se houver uma classificação ao qual é referente (MONTEIRO (2002)).

Após esta análise, conclui-se que a entidade Anestesia pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um Tipo.

5.1.2.3 Prescrição solução

É o modelo especializado no processo de cura sugerido pelo médico ao paciente. Abrangem os medicamentos e suas respectivas posologias, médicos que os prescrevem, enfermeiras que ministram os medicamentos e os pacientes que recebem esse tratamento. Dele, podemos encontrar quais as combinações de medicamentos e posologias estão funcionando. A princípio, foram formadas as classes das combinações conhecidas e executado o Naive Bayes para classificá-las. Em seguida, foram analisadas as que não se encaixam nos padrões pré-estabelecidos, mas que são efetivamente satisfatórias.

É um modelo voltado a auxiliar na descoberta de novos tratamentos, mais eficientes em relação ao custo e ao tempo de cura.

Para a construção do diagrama entidade-relacionamento, foram aplicados o primeiro e o segundo passo, mostrados na figura 35 .

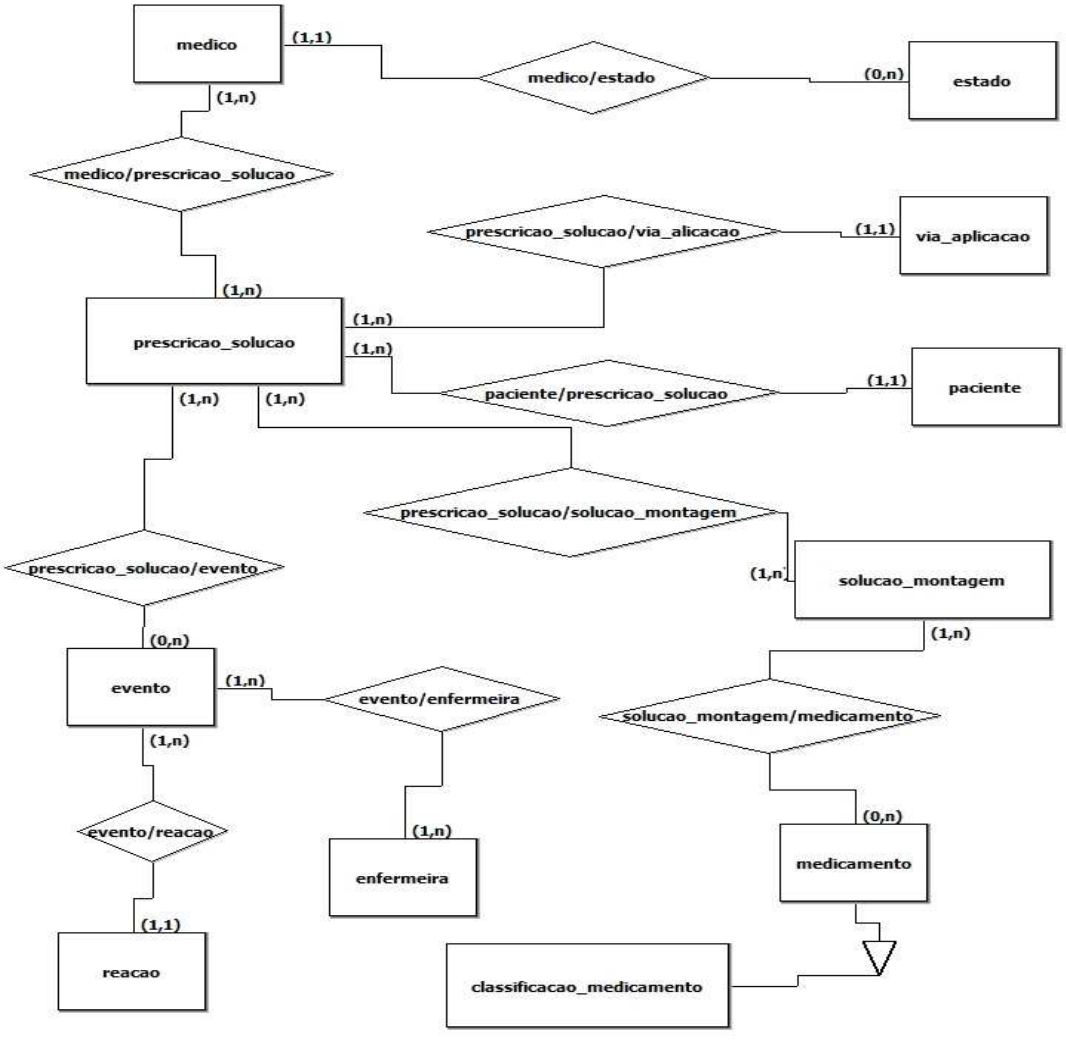

Figura 35: Representação de entidades: *Prescrição Solução*

Para a construção do modelo entidade-relacionamento, após a construção do diagrama entidade-relacionamento, foram inseridos os atributos e excluídas as tabelas que não contêm dado e não têm utilização pelo sexto passo

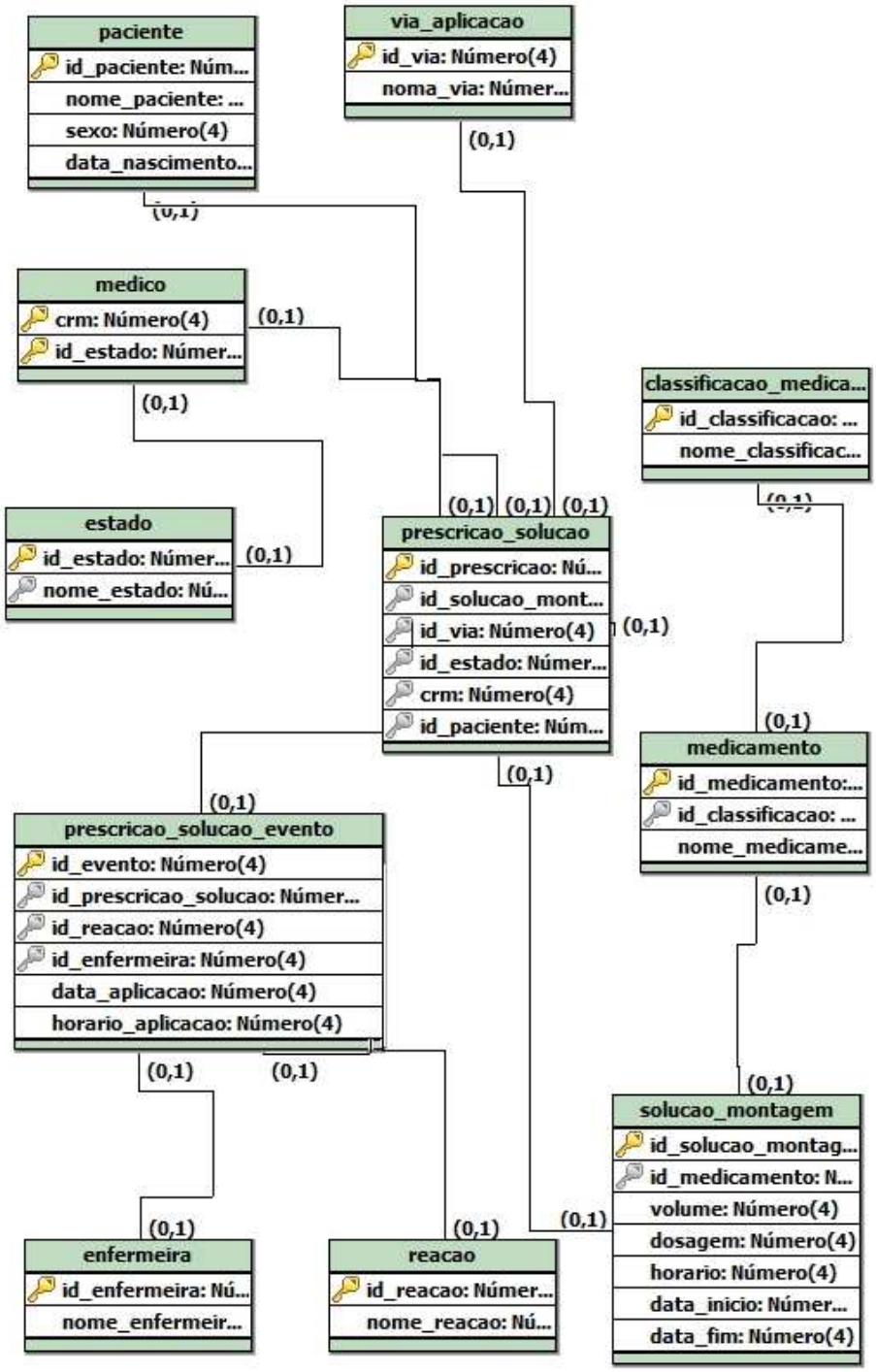

Figura 36: Modelo Lógico: Prescrição Solução

A Tabela 14 representa as entidades, com suas respectivas chaves primárias e atributos correspondentes, e se estas são algum subtipo ou são principais, denominadas de raiz, após a aplicação do primeiro passo.

Tabela 17: Representação de entidades: *prescrição solução*

Entidade	**Chave Primária**	**Atributos**
solucao_montagem	id_solucao_ montagem	id_medicamento volume dosagem horario data_inicio data_fim
vias_aplicacao	id_via	nome_via
prescricao_solucao	id_precricao_ solucao	id_solucao_ montagem id_via id_estado id_crm id_paciente
Reação	id_reacao	nome_reacao
Enfermeira	id_enfermeira	nome_enfermeira
prescricao_solucao_evento	id_evento	id_precricao_ solucao id_reacao id_enfermeira data_aplicacao hora_aplicacao

A Tabela 15 representa as entidades e seus relacionamentos, bem como a dependência identificada como imagem, as cardinalidades entre as entidades, obtidas pelo segundo passo.

Tabela 18: Representação dos relacionamentos: *prescrição solução*

Relações	**Domínio**	**Imagem**	**Cardinalidade**	
			Domínio	**Imagem**
estado/medico	estado	medico	(0,N)	(1,1)
prescricao/via[2]	prescricao	via	(1,N)	(1,1)
via/prescricao	via	prescricao	(1,1)	(1,N)
paciente/prescricao	paciente	prescricao	(1,N)	(1,1)
prescricao/solucao[3]	prescricao	solucao	(1,N)	(1,N)
solucao/prescricao	solucao	prescricao	(1,N)	(1,N)
solucao/ medicamento	solucao	medicamento	(1,N)	(0,N)
medicamento/ solucao	medicamento	solucao	(0,N)	(1,N)
prescricao/evento	prescricao	evento	(1,N)	(0,N)
evento/prescricao	evento	prescricao	(0,N)	(1,N)
evento/reacao	evento	reacao	(1,N)	(1,1)
reacao/evento	reacao	evento	(1,1)	(1,N)
evento/enfermeira	evento	enfermeira	(1,N)	(1,N)
enfermeira/evento	enfermeira	evento	(1,N)	(1,N)

A Tabela 16 representa o conceito dos metadados, com descrição e entidades respectivamente, obtidas pelo quarto passo.

Tabela 19: Documentações dos conceitos: *prescrição solução (parte 1)*

Nome do conceito	SOLUCAO_MONTAGEM
Descrição do conceito	Conjunto de técnica, equipamento, medicamento, quantidade e período usado para a aplicação de um medicamento.
Sinônimo	Composição.
Nome do conceito	VIAS_APLICACAO
Descrição do conceito	Forma pela qual a solução deve ser aplicada (injetada, inalada e outros)
Sinônimo	-

Tabela 20: Documentações dos conceitos: *prescrição solução (parte 2)*

Nome do conceito	PRESCRICAO_SOLUCAO
Descrição do conceito	União de todas as tarefas para a aplicação da solução no paciente.
Sinônimo	-
Nome do conceito	REACAO
Descrição do conceito	Efeitos ocorridos, decorrentes da aplicação da solução.
Sinônimo	Efeito colateral.
Nome do conceito	ENFERMEIRA
Descrição do conceito	É um componente responsável pela aplicação dos medicamentos prescritos pelos médicos.
Sinônimo	Profissional de saúde (não confundir com médicos).
Nome do conceito	PRESCRICAO_SOLUCAO_EVENTO
Descrição do conceito	É a união de acontecimentos envolvidos durante a aplicação de um medicamento.
Sinônimo	-

A Tabela 17 representa as propriedades, com nome do atributo, tipo de dados (imagem), entidade a que pertence denominado domínio, tipo que é definido por datatype (tipo de dados) ou object (associação entre entidades) e pela característica que denota se é um atributo funcional (necessário) ou não, obtidas através do terceiro passo, quinto passo, sexto passo e sétimo passo.

Tabela 21: Atributos transformados em propriedades: *datatype e object (prescrição solução) (parte 1)*

Nome	**Tipo**	**Domínio**	**Imagem**	**Característica**
Paciente/ Prescricao_solucao	Object	PACIENTE	-	Funcional
Medico/ Prescricao_solucao	Object	MEDICO	-	Funcional
Id_solucao_montagem	Datatype	SOLUCAO_MONTAGEM	Inteiro	Funcional
Id_medicamento	Datatype	SOLUCAO_MONTAGEM	Inteiro	Funcional
Volume	Datatype	SOLUCAO_MONTAGEM	Decimal	Funcional
Dosagem	Datatype	SOLUCAO_MONTAGEM	Decimal	Funcional
Horário	Datatype	SOLUCAO_MONTAGEM	Hora	Funcional

Tabela 22:: Atributos transformados em propriedades: *datatype e objeto (prescrição solução) (parte 2)*

Data_inicio	Datatype	SOLUCAO_MONTAGEM	Data	Funcional
Data_fim	Datatype	SOLUCAO_MONTAGEM	Data	Funcional
Solucao_montagem/ Medicamento	Object	SOLUCAO_MONTAGEM	-	Funcional
Id_via	Datatype	VIAS_APLICACAO	Inteiro	Funcional
Nome_via	Datatype	VIAS_APLICACAO	Texto	Funcional
Id_prescricao_solucao	Datatype	PRESCRICAO_SOLUCAO	Inteiro	Funcional
Id_solucao_montagem	Datatype	PRESCRICAO_SOLUCAO	Inteiro	Funcional
Id_via	Datatype	PRESCRICAO_SOLUCAO	Inteiro	Funcional
Id_estado	Datatype	PRESCRICAO_SOLUCAO	Inteiro	Funcional
Id_crm	Datatype	PRESCRICAO_SOLUCAO	Inteiro	Funcional
Id_paciente	Datatype	PRESCRICAO_SOLUCAO	Inteiro	Funcional
Prescricao_solucao/ Via_aplicacao	Object	PRESCRICAO_SOLUCAO	-	Funcional
Prescricao_solucao/ Solucao_montagem	Object	PRESCRICAO_SOLUCAO	-	Funcional
Prescricao_solucao/ Evento	Object	PRESCRICAO_SOLUCAO	-	Funcional
Id_reacao	Datatype	REACAO	Inteiro	Funcional
Nome_reacao	Datatype	REACAO	Texto	Funcional
Id_enfermeira	Datatype	ENFERMEIRA	Inteiro	Funcional
Nome_enfermeira	Datatype	ENFERMEIRA	Texto	Funcional
Id_evento	Datatype	PRESCRICAO_SOLUCAO_EVENTO	Inteiro	Funcional
Id_precricao_solucao	Datatype	PRESCRICAO_SOLUCAO_EVENTO	Inteiro	Funcional
Id_reacao	Datatype	PRESCRICAO_SOLUCAO_EVENTO	Inteiro	Funcional
Id_enfermeira	Datatype	PRESCRICAO_SOLUCAO_EVENTO	Inteiro	Funcional
Data_aplicacao	Datatype	PRESCRICAO_SOLUCAO_EVENTO	Data	Funcional
Hora_aplicacao	Datatype	PRESCRICAO_SOLUCAO_EVENTO	Hora	Funcional
Evento/Reacao	Object	PRESCRICAO_SOLUCAO_EVENTO	-	Funcional
Evento/Enfermeira	Object	PRESCRICAO_SOLUCAO_EVENTO	-	Funcional

A Tabela 18 representa as restrições entre as entidades, demonstrando o nome do conceito, tipo de restrição (se deve ter algum valor), tipo de condição (se é necessário, suficiente ou ambos) e notação, que denota a relação entre as entidades, que representam o oitavo e nono passos.

Tabela 23: Representação das restrições: *prescrição solução*

Nome do Conceito	**Tipo da Restrição**	**Tipo da Condição**	**Notação**
PRESCRICAO_SOLUCAO	SOME VALUES OF	Necessária	PRESCRICAO_SOLUCAO: prescricao_solucao/medico.medico
PRESCRICAO_SOLUCAO	ALL VALUES OF	Necessária	PRESCRICAO_SOLUCAO: prescricao_solucao/via_aplicacao. via_aplicacao
VIA_APLICACAO	SOME VALUES OF	Necessária	VIA_APLICACAO: prescricao_solucao/via_aplicacao. prescricao_solucao
PRESCRICAO_SOLUCAO	SOME VALUES OF	Necessária	PRESCRICAO_MEDICA: paciente/ prescricao_solucao.paciente
PRESCRICAO_SOLUCAO	SOME VALUES OF	Necessária	PRESCRICAO_SOLUCAO: prescricao_solucao/ evento.evento
EVENTO	SOME VALUES OF	Necessária	PRESCRICAO_SOLUCAO: prescricao_solucao/ evento.prescricao_solucao
EVENTO	SOME VALUES OF	Necessária	PRESCRICAO_SOLUCAO: evento/enfermeira.enfermeira
ENFERMEIRA	SOME VALUES OF	Necessária	PRESCRICAO_SOLUCAO: evento/enfermeira.evento
EVENTO	INTER-SECTION OF	Necessária	EVENTO: evento/ reacao.reacao EVENTO: evento / reacao.reacao
REACAO	SOME VALUES OF	Necessária	REACAO: evento/reacao.evento
PRESCRICAO_SOLUCAO	SOME VALUES OF	Necessária	PRESCRICAO_SOLUCAO: prescricao_solucao/ solucao_montagem. solucao_montagem
SOLUCAO_MONTAGEM	SOME VALUES OF	Necessária	SOLUCAO_MONTAGEM: prescricao_solucao/ solucao_montagem.prescricao_solucao
SOLUCAO_MONTAGEM	SOME VALUES OF	Necessária	SOLUCAO_MONTAGEM: prescricao_solucao/ medicamento.medicamento

5.1.3 Atribuição das metapropriedades às entidades do domínio prescrição solução

5.1.3.1 Solução_Montagem

A entidade Solução_Montagem armazena informações sobre conjunto de técnica, equipamento, medicamento, quantidade e período usado para a aplicação de um medicamento.

A seguir, é realizada uma análise da entidade Solução_Montagem, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – toda Solução_Montagem será necessariamente feita em um paciente.
- Não fornece CI (-O) – desde que uma mesma pessoa possa receber prescrição de medicamentos diferentes vezes, uma condição de identidade (CI) é fornecida por Solução_Montagem.
- Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.
- Dependente (+D) – Solução_Prescrição será considerada externamente dependente de medicamentos e médico. Fornecido por MONTEIRO (2002), alguém somente poderá receber medicamentos de uma determinada Solução_Montagem se houver recebido alguma prescrição médica da mesma.

Após esta análise, conclui-se que a entidade Solução_Montagem pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.3.2 Vias_Aplicação

A entidade Vias_Aplicação armazena informações sobre a forma que a solução deve ser aplicada (injetada, inalada e outros).

A seguir, é realizada uma análise da entidade Vias_Aplicação, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – Vias_Aplicação existirá durante toda a sua existência.
- Não fornece CI (-O) – nenhuma Vias_Aplicação pode ser identificada globalmente por meio de uma característica própria.
- Não executa CI (-I) – executa uma CI relacionada a Prescrição_Solução, de um

modo geral.

• Dependente (+D) – é externamente vinculada a Prescrição_Solução e somente irá existir se houver uma Prescrição_Solução ao qual é referente, fornecido por MONTEIRO (2002).

Após esta análise, conclui-se que a entidade Vias_Aplicação icação pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Tipo*.

5.1.3.3 Prescrição_Solução

A entidade Prescrição_Solução armazena informações sobre a união de todas as tarefas para a aplicação da solução no paciente.

A seguir, é realizada uma análise da entidade Prescrição_Solução, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirrígida (+R) – toda Prescrição_Solução será necessariamente feita em um paciente.

• Não fornece CI (-O) – desde que uma mesma pessoa pode receber prescrição de medicamentos diferentes vezes, uma condição de identidade (CI) é fornecida por Prescrição_Solução.

• Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.

• Dependente (+D) – Prescrição_Solução será considerada externamente dependente de medicamentos, médico e paciente. Fornecido por MONTEIRO (2002), alguém somente poderá receber medicamentos de uma determinada Prescrição_Solução se houver recebido alguma prescrição médica da mesma.

Após esta análise, conclui-se que a entidade Prescrição_Solução pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.3.4 Reação

A entidade Reação armazena informações sobre os efeitos ocorridos decorrentes da aplicação da solução.

A seguir, é realizada uma análise da entidade Reação, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirígida (+R) – Reação existirá durante toda a sua existência.

• Não fornece CI (-O) – nenhuma reação pode ser identificada globalmente por meio de uma característica própria.

• Não executa CI (+I) – executa uma CI relacionada à Prescrição_Solução, de um modo geral.

• Dependente (+D) – é externamente vinculada a Prescrição_Solução e somente irá existir se houver uma Prescrição_Solução ao qual é referente, fornecido por MONTEIRO (2002),

Após esta análise, conclui-se que a entidade Reação pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Tipo*.

5.1.3.5 Enfermeira

A entidade Enfermeira armazena informações sobre o componente responsável pela aplicação dos medicamentos prescritos pelos médicos

A seguir, é realizada uma análise da entidade Enfermeira, com relação a cada uma das metapropriedades vistas anteriormente:

• Antirrígida (+R) – Enfermeira existirá durante toda a sua existência.

• Não fornece CI (-O) – nenhuma enfermeira pode ser identificada globalmente por meio de uma característica própria.

• Não executa CI (+I) – executa uma CI relacionada à Prescrição_Solução_Evento, de um modo geral.

• Dependente (+D) – é externamente vinculada a Prescrição_Solução_Evento e somente irá existir se houver uma Prescrição_Solução_Evento ao qual é referente, fornecido por MONTEIRO (2002).

Após esta análise, conclui-se que a entidade Enfermeira pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Tipo*.

5.1.3.6 Prescrição_Solução_Evento

A entidade Prescrição_Solução_Evento armazena informações sobre a união de

acontecimentos envolvidos durante a aplicação de um medicamento. A seguir, é realizada uma análise da entidade Prescrição_Solução_Evento, com relação a cada uma das metapropriedades vistas anteriormente:

- Antirrígida (+R) – toda Prescrição_Solução_Evento será necessariamente feita em um paciente.
- Não fornece CI (-O) – desde que uma mesma pessoa possa receber prescrição de medicamentos diferentes vezes, uma condição de identidade (CI) é fornecida por Prescrição_Solução_Evento.
- Executa CI (+I) – executa uma CI relacionada à pessoa, de um modo geral.
- Dependente (+D) – Prescrição_Solução_Evento será considerada externamente dependente de medicamentos, médico e paciente. Ffornecido por MONTEIRO (2002), alguém somente poderá receber medicamentos de uma determinada Prescrição_Solução_Evento se houver recebido alguma prescrição médica da mesma.

Após esta análise, conclui-se que a entidade Prescrição_Solução_Evento pode ser classificada, de acordo com GUARINO e WELTY (2000a), como um *Papel Material*.

5.1.4 Aplicação de Mineração

A intenção dos modelos conceituais e lógicos gerados é promover a aplicação de mineração de dados. A ideia principal, num primeiro momento, era verificar a possibilidade de resultados que pudessem direcionar, dentro das áreas escolhidas de domínio, a extração de conhecimento. Entretanto, a complexidade é muito grande e não foi possível uma definição mais precisa dos atributos do domínio, pois o sistema utilizado pelo Hospital não oferece dados suficientes em relação a exames, nos quais são definidos todos os referenciais em relação às prescrições descritas no trabalho.

A mineração de dados ficou limitada pelas características do modelo a uma análise de um cenário possível e à simulação de outros cenário.

5.1.4.1 Cenário 1 de mineração de dados

Neste cenário, de acordo com os modelos gerados de prescrição atendimento, prescrição médica e prescrição solução, foram selecionados os seguintes atributos: proteínas, idade, lipídios, carboidratos, sexo e peso, e foi escolhido como classificador os pacientes que têm ou não problemas de fígado. Este fato interfere diretamente na produção de proteínas e verifica o comportamento dos demais atributos na classificação.

Tabela 24: Mineração – Resultados do Cenário 1

Algoritmo	**Tipo**	**Instâncias classificadas corretamente**
Naive Bayes	CLASSIFICAÇÃO	73.7621 %
Decision Table	CLASSIFICAÇÃO	74.6747 %
PART	CLASSIFICAÇÃO	76.3531 %
J48	CLASSIFICAÇÃO	82.7543 %
NaiveBayesUpdatealbe	CLASSIFICAÇÃO	74.7752 %
MultilayerPerceptron	CLASSIFICAÇÃO	76.8623 %

```
NaiveBayesUpdatealbe

=== Classifier model (full training set) ===

Naive Bayes Classifier

Class 0: Prior probability = 0.65

proteinas:  Normal Distribution. Mean = 109.9541 StandardDev = 26.1114 WeightSum = 21278 Precision =
1.474074074074074
lipidio:  Normal Distribution. Mean = 68.1397 StandardDev = 17.9834 WeightSum = 21278 Precision =
2.652173913043478
sexo:  Normal Distribution. Mean = 19.8356 StandardDev = 14.8974 WeightSum = 21278 Precision = 1.98
carboidrato:  Normal Distribution. Mean = 68.8507 StandardDev = 98.828 WeightSum = 21278 Precision =
4.572972972972973
peso:  Normal Distribution. Mean = 30.3009 StandardDev = 7.6833 WeightSum = 21278 Precision =
0.2716599190283401
idade:  Normal Distribution. Mean = 31.2494 StandardDev = 11.6059 WeightSum = 21278 Precision =
1.1764705882352942

Class 1: Prior probability = 0.35

proteinas:  Normal Distribution. Mean = 141.2581 StandardDev = 31.8728 WeightSum = 6608 Precision =
1.474074074074074
lipidio:  Normal Distribution. Mean = 70.718 StandardDev = 21.4094 WeightSum = 6608 Precision =
2.652173913043478
sexo:  Normal Distribution. Mean = 22.2824 StandardDev = 17.6992 WeightSum = 6608 Precision = 1.98
carboidrato:  Normal Distribution. Mean = 100.2812 StandardDev = 138.4883 WeightSum = 268 Precision =
4.572972972972973
peso:  Normal Distribution. Mean = 35.1475 StandardDev = 7.2537 WeightSum = 6608 Precision =
0.2716599190283401
idade:  Normal Distribution. Mean = 37.0808 StandardDev = 10.9146 WeightSum = 6608 Precision =
1.1764705882352942

Time taken to build model: 450.01 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances       21278               76.3021 %
Incorrectly Classified Instances      6608               23.6979 %
Kappa statistic                          0.4674
Mean absolute error                      0.2811
Root mean squared error                  0.4133
Relative absolute error                 61.8486 %
Root relative squared error             86.7082 %
Total Number of Instances            27886

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   Class
                 0.842    0.384      0.803     0.842     0.822    0
                 0.616    0.158      0.676     0.616     0.645    1

=== Confusion Matrix ===

   a   b   <-- classified as
 15337  2842 |   a = 0
 3766 5941 |   b = 1
```

Figura 37: resultado da execução do algoritmo Naive Bayes

```
J48
=== Classifier model (full training set) ===
J48 pruned tree
------------------

proteinas <= 127
|   peso <= 26.4: 0 (132.0/3.0)
|   peso > 26.4
|   |   idade <= 28: 0 (180.0/22.0)
|   |   idade > 28
|   |   |   proteinas <= 99: 0 (55.0/10.0)
|   |   |   proteinas > 99
|   |   |   |   lipidio <= 0.561: 0 (84.0/34.0)
|   |   |   |   lipidio > 0.561
|   |   |   |   |   carboidrato <= 6
|   |   |   |   |   |   idade <= 30: 1 (4.0)
|   |   |   |   |   |   idade > 30
|   |   |   |   |   |   |   idade <= 34: 0 (7.0/1.0)
|   |   |   |   |   |   |   idade > 34
|   |   |   |   |   |   |   |   peso <= 33.1: 1 (6.0)
|   |   |   |   |   |   |   |   peso > 33.1: 0 (4.0/1.0)
|   |   |   |   |   lipidio > 6: 1 (13.0)
proteinas > 127
|   peso <= 29.9
|   |   proteinas <= 145: 0 (41.0/6.0)
|   |   proteinas > 145
|   |   |   idade <= 25: 0 (4.0)
|   |   |   idade > 25
|   |   |   |   idade <= 61
|   |   |   |   |   peso <= 27.1: 1 (12.0/1.0)
|   |   |   |   |   peso > 27.1
|   |   |   |   |   |   lipidio <= 0.396: 1 (8.0/1.0)
|   |   |   |   |   |   lipidio > 0.396: 0 (3.0)
|   |   |   |   idade > 61: 0 (4.0)
|   peso > 29.9
|   |   proteinas <= 157
|   |   |   idade <= 30: 0 (40.0/13.0)
|   |   |   idade > 30: 1 (60.0/17.0)
|   |   proteinas > 157: 1 (92.0/12.0)

Number of Leaves  :   20

Size of the tree :      39

Time taken to build model: 659.03 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances       23456               84.1146 %
Incorrectly Classified Instances      4430               15.8854 %
Kappa statistic                          0.6319
Mean absolute error                      0.2383
Root mean squared error                  0.3452
Relative absolute error                 52.4339 %
Root relative squared error             72.4207 %
Total Number of Instances            27886

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.936    0.336      0.839     0.936     0.885    0
  0.664    0.064      0.848     0.664     0.745    1

=== Confusion Matrix ===

   a   b   <-- classified as
 16888  1152 |   a = 0
  3278  6568 |   b = 1
```

Figura 38: resultado da execução do algoritmo J48

Resultados Obtidos Algoritmo de Classificação

Os valores obtidos com os algoritmos de classificação da Tabela 24 representam o resultado final da classificação para pacientes que têm problema de fígado. As figuras 37 e 38 mostram respectivamente a execução dos algoritmos Naive Bayes e J48. O resultado obtido permite ao especialista perceber a quantidade menor de proteínas, mas não é um fator preponderante, pois a idade também faz com que haja menor produção delas. Outra análise, realizada com o auxílio do Dr. Angelmar Roman, é que estes atributos não possuem diferenciação entre homens e mulheres, o que seria determinante pela quantidade de hormônios, e que seria mais conclusivo o resultado em relação aos demais atributos, como lipídios, carboidratos e peso.

5.1.4.2 Cenário 2 de mineração de dados - Simulação

Neste cenário, de acordo com os modelos gerados de prescrição atendimento, prescrição médica e prescrição solução, foram selecionados outros atributos que não pertencem aos modelos mas que podem determinar de maneira mais conclusiva os resultados da mineração. Foram utilizados os seguintes atributos: função renal, taxa de gordura, dosagem hormonal, idade e peso; e como classificador foi escolhido novamente os pacientes que têm ou não problemas de fígado, pois este fator interfere diretamente na produção de proteínas e verifica o comportamento dos demais atributos na classificação.

Os valores obtidos com os algoritmos de classificação da tabela 25 representam o resultado final da classificação para pacientes que têm problema de fígado. Os atributos de função renal estão ligados à produção de proteínas e à taxa de gordura são maiores nas mulheres, independente da idade. Os hormônios são determinantes para a análise de proteínas, lipídios e demais atributos do cenário 1. Entretanto, o Dr. Angelmar Roman alerta que estes casos devem ser analisados em conjunto com os referenciais dos exames de hemograma ou exames correlatos, nos quais os valores de refência são preponderantes .

Tabela 25: Mineração – Resultados do Cenário 2

Algoritmo	**Tipo**	**Instâncias classificadas corretamente**
Naive Bayes	CLASSIFICAÇÃO	72.7731 %
Decision Table	CLASSIFICAÇÃO	73.6848 %
PART	CLASSIFICAÇÃO	75.3431 %
J48	CLASSIFICAÇÃO	82.7643 %
NaiveBayesUpdatealbe	CLASSIFICAÇÃO	75.5753 %
MultilayerPerceptron	CLASSIFICAÇÃO	76.8623 %

5.2 Discussão

Os resultados provenientes da aplicação da metodologia proposta para construção de modelos conceituais foram analisados em conjunto por profissionais da área de TI do hospital e pelo médico Dr. Angelmar Roman.

A primeira análise foi através do modelo conceitual na compreensão do domínio. A segunda análise foi através da validação das metapropriedades. A terceira foi através das possíveis utilizações dos modelos para oferecer extração de conhecimento através de consultas complexas ou mineração de dados. A validação é a própria compreensão dos modelos e os resultados obtidos através dos cenários de mineração.

A apresentação dos resultados foi realizada através dos modelos conceituais e

lógicos desenvolvidos utilizando a metodologia proposta, mostrados nas Figuras 32, 34 e 36 (modelos conceituais) e nas Figuras 33, 35 e 37 (modelos lógicos).

A extensão do modelo original dificultou a análise por parte dos especialistas consultados, pela quantidade e tabelas, em torno de 2.700, e pela falta de documentação necessária para levantar os processos, promover entrevistas com diversos profissionais das áreas envolvidas, gerar novas bases para fazer os testes, devido ao alto número de dados existentes, definir de forma clara os domínios a serem estudados, compreender as noções subjacentes às metapropriedades ontológicas atribuídas às entidades. Tal fato é totalmente compreensível, uma vez que a análise ontológica constitui-se normalmente numa novidade para modeladores conceituais, sendo que os conceitos relacionados a esta análise precisam ainda ser mais profundamente estudados e assimilados por estes profissionais. O Dr. Angelmar Roman auxilou de forma decisiva a tornar clara a definição das metapropriedades e o escopo dos modelos conceituais, visto o seu conhecimento na área .

A aplicação da metodologia promoveu uma percepção mais clara do problema para os modeladores, uma vez que eles puderam entender melhor a essência dos conceitos presentes no domínio a ser modelado. Dessa forma, os modelos produzidos tendem naturalmente a retratar melhor e de forma mais clara a realidade dos domínios e a necessidade de conhecer o que os modelos podem oferecer em termos de extração de conhecimento, quer seja pela forma de consultas complexas, quer seja pela aplicação de mineração de dados.

Este capítulo abordou a utilização da metodologia proposta na construção de um modelo para mineração de dados, voltada fortemente à análise e que combina o aspecto da utilização da ontologia como fator preponderante de validação do modelo lógico.

Até então, nenhum editor ou metodologia de ontologia visto no capítulo 2 fez uma busca na base de dados, trazendo uma compreensão suficiente para representar um modelo de forma a priorizar as metapropriedades, a fim de tornar o modelo de dados compreensível e utilizá-lo para um suporte a mineração de dados.

É possível uma mineração de dados a partir de uma proposição de aplicação da ontologia. Foi usada a Ontoclean (GUARINO e WELTY, 2002), que permite validar o modelo através da rigidez, dependência, unidade e identidade e que permite validar as entidades a participarem do modelo e dos seus relacionamentos.

Retomando as perguntas propostas no objetivo do trabalho:

1) É possível uma extensão das metodologias existentes de ontologia para mineração? A resposta é sim, mas com validação formal da construção do modelo.

A proposta possibilita dentro de um domínio apresentar modelos conceituais antes não conhecidos e fornecer as condições de compreensão sobre os domínios e atributos . Neste momento somente o especialista da área pode direcionar a proposição do trabalho, num segundo momento com regras a serem definidas neste domínio, o trabalho ficaria sem a dependência do mesmo

Existem vários trabalhos, metodologias e reconhecidas, mas a importância de resgatar o que significa os modelos conceituais a partir do modelo físico sem auxílio de documentação e trabalhando somente com os dados e observação do comportamento do sistema de informação como um todo, não pode ser observado nos trabalhos, que tratam conceitos de topo e a partir daí mapeiam classes no caso de orientação a objetos e posteriormente tratam o banco.

2) O modelo gerado, através da metodologia proposta, é passível de ser aplicado a diversos algoritmos de mineração de dados? A resposta é sim, mas, neste trabalho, a procura intensa foi validar um modelo que suportasse mineração e explorar algoritmos de classificação.

Muitos resultados foram omitidos deste trabalho, por estratégia do Hospital Cruz Vermelha, e não puderam ser mostrados.

Os trabalhos com acesso a base de dados foram feitos sempre no local, com supervisão dos profissionais da área de informática, por questões de segurança e sigilo das informações.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho faz referência aos conhecimentos ontológicos, utilizados de diversas maneiras em conjunto com conhecimentos relacionados à modelagem conceitual, e que se têm mostrado importantes. Primeiramente, foi visto que uma análise ontológica de construtores utilizados em modelagem conceitual, por meio de ontologias de nível topo, pode ser útil para ampliar a semântica embutida em modelos conceituais de sistemas de informação. Em outro aspecto, viu-se que a utilização de ontologia de

domínio, como ferramenta de apoio ao projetista em modelagem conceitual de banco de dados, pode ser valiosa, uma vez que traz conhecimentos sólidos do domínio a ser modelado.

Ainda buscando interagir conhecimentos relacionados à ontologia e à modelagem conceitual, a técnica apresentada por este trabalho proporciona a validação de um modelo conceitual, por meio de uma análise ontológica das entidades, que representam elementos do domínio. Para possibilitar tal validação, foi realizado um mapeamento dos tipos de propriedades, vistos no Capítulo 4, em elementos do modelo conceitual. Este mapeamento permite que a ontologia seja aplicada a relacionamentos de Generalização e Especialização presentes em modelos conceituais, se existirem com a mesma validação utilizada anteriormente.

Como contribuição, pôde-se verificar que o modelo validado, obtido a partir da aplicação da técnica em questão, representa melhor a realidade do domínio, diminuindo risco de inconsistências, uma vez que elimina a presença de informações redundantes em outras entidades que apresentem características em comum.

Dessa forma, é possível concluir que a metodologia proposta contribui para a obtenção de informações precisas sobre o domínio a ser modelado, uma vez que força o modelador a separar informações estáveis, representando propriedades rígidas do domínio, de informações não estáveis, na fase de modelagem. Isto conduz a modelos conceituais mais claros e consistentes, levando, consequentemente, a implementações de sistemas mais fáceis de serem integrados e mantidos.

Outra contribuição importante foi a mudança de estratégia da empresa que fornece o sistema para o hospital, mudando o aspecto ambiental do sistema e aumentando a visibilidade de conceitos abordados pelo sistema.

Os resultados de mineração não passaram por avaliação qualitativa aprofundada, mas fornecem subsídios que, a partir da modelagem conceitual dos domínios com clareza e abrangência definida para cada área de atuação, é possível obter resultados relevantes

Como trabalhos futuros, abre uma nova perspectiva de implementação e tratamento dinâmico em bases de dados legados e validações de vários outros algoritmos, para que os sistemas tenham um poder decisório maior e mais preciso. Já existe uma ferramenta sendo desenvolvida para aplicar esta metodologia e está na fase de construção.

REFERÊNCIAS BIBLIOGRÁFICAS

http://www.algo.be/en.html

http://www.algo.be/cl/KMgen/download.htm

http://semanticwiki-en.saltlux.com/index.php/OntoStudio

http://osm.cs.byu.edu/CS652s09/papers/Weiten.OntoSTUDIO.pdf

http://emeld.org/workshop/2005/papers/mostowfi-paper.html

AHA D.; KIBLER D. e ALBERT M.: Instance-Based Learning

Algorithms. Kluwer Academic Publishers, 1991. 6, 37-66

ALMEIDA, M; BAX, M. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. Revista Ciência da Informação, 32(3), 2003.

ARPÍREZ, J. C; CORCHO, O; FERNÁNDEZ-LÓPEZ, M; GÓMEZ-PÉREZ, A. Webode:a scalable workbench for ontological engineering. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE CAPTURE. Victoria, British Columbia, Canada, 2001.

AZEVEDO FILHO, A. Princípios de Inferência Dedutiva e Indutiva: Noções de Lógica e Métodos de Prova. 1ª Edição 2010, Scotts Valley.

BAADER, F., CALVANESE, D., MCGUINNESS, D., NARDI, D. e PATEL-SCHNEIDER, P. TheDescription Logic Handbook. Theory, Implementation and Applications, Cambridge, 2003

BASSERVILLE, M.; NIKIFOROV, I. V. Detection of Abrupt Changes:Theory and Application, Englewoods Cliffs - NJ, Prentice Hall, 1993.

BECHHOFER, S. et al. OILEd:a reasonable ontology editor for the semantic web. In KI2001, Joint German/Austrian conference on Artificial Intelligence, volume LNAI Vol. 2174, pp. 396-408, Vienna, 2001.

BERNARAS, A.; LARESGOITI, I.; CORERA, J. Building and Reusing Ontologies for Electrical Network Applications. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE, ECAI/96, p. 298-302, 1996.

BECHHOFER, S.; HORROCKS, I.; GOBLE, C., STEVENS, R.: OilEd: a reasonable ontology editor for the Semantic Web. Proceedings of KI2001, JointGerman/Austrian Conference on Artificial Intelligence (2001)

BERNERS-LEE, T; HENDLER, J; O., L.*The semantic web.Scientific American*, 05 2001.

BERNSTEIN, A., Hill, S., & Provost, F. (2001). *Towards intelligent assistance for the data mining process:An ontology-based approach*. CeDER *Working Paper* IS-02-02, *New York University.*

BERNSTEIN, A., Provost, F., & Hill, S. (2005). *Towards intelligent assistance for the data mining process: An ontology-based approach for cost/sensitive classification. In IEEE Transactions on Knowledge and Data Engineering, 17*(4), 503-518.

BOGORNY, V., Engel, P. M., & Alvares, L.O. (2005). Towards the reduction of spatial join for knowledge discovery in geographic databases using geo-ontologies and spatial integrity constraints. In M.Ackermann, B. Berendt, M. Grobelink, & V. Avatek (Eds.), *Proceedings ECML/PKDD SecondWorkshop on Knowledge Discovery and Ontologies* (pp. 51-58).

BOUNIF, H., Spaccapietra, S., & Pottinger, R. (2006, September 12-15). *Requirements ontology and multirepresentation strategy for database schema evolution.Paper presented at the 2nd VLDB Workshop on Ontologies-based techniques for Databases and Information Systems*. Seoul, Korea.

BOHRING, H. e AUER, S. Mapping XML to OWL Ontologies. Marktplatz Internet: Von e-Learning bis e-Payment. Leipziger Informatik-Tage (LIT2005), Leipzig, Germany, pp.147-156, 2005.

BORTOLETO, S.; EBECKEN, N. F. F.: "Ontology Model for Multi-relational Data Mining Application," isda, vol. 2, pp.460-463, 2008 Eighth International Conference on Intelligent Systems Design and Applications, 2008

BRAZDIL , P.B.; SOARES, C.; COSTA, J.P.: Ranking de algoritmos de aprendizagem: IBL Uso e Meta-Aprendizado em Tempo e Resultados Precisão, 2001.

BRAY T; PAOLI, J; SPERBERG-MCQUEEN, C; MALER, E; YERGEAU, F. Extensible markup language (xml) *1.0.*Disponível em

<http://www.w3.org/TR/2004/REC-xml-20040204/>. Acessado em julho/2008.

BREIMAN, L.; FRIEDMAN, J.H.; OLSHEN, R.A.; STONE, C.J.Classification and Regression Trees, Belmont, California: Wadsworth, 1984.

BREITMAN, K. Web Semântica:A Internet do Futuro. Brasil, LTC, 212P, 2005.

BREZANY, P., Janciak, I., Woehrer, A., & Tjoa, A.M. (2004).*GridMiner: A framework for knowledge discovery on the Grid from a vision to design and implementation*. Cracow Grid Workshop. Cracow, Poland: Springer.

BRIDEWELLl, W., Sánchez, J. N., Langley, P., & Billwen, D. (2006).An Interactive environment for the modeling on discovery of scientific knowledge.*International Journal of Human-Computer Studies*, 64, 1009-1014.

BRIEN P.; ZHAN, C.; Domain Ontology Management Environment, Proceedings of 33rd Hawaii International Conference on Systems Sciences, Jan/2000, IEEE Computer Sciences pg.9, Vol. I.

CALIARI, F.B; DERONTO: Método para construção de ontologias a partir de diagramas entidade-relacionamento (tese de dissertação de mestrado, universidade Tecnológica Federal do Paraná), 2007.

CANNATARO, M., & Comito, C. (2003, May 20-24). *A data mining ontology for Grid programming*. Paper presented at the I Workshop on Semantics Peer to Peer and Grid Computing. Budapest. Acessado em Agosto, 2008, from http://www.isi.edu/~stefan/SemPGRID

CANNATARO, M., Congiusta, A. Pugliese, A., Talia, D., & Trunfio, P. (2004). Distributed data mining on Grids: Services, tools, and applications.*IEEE Transactions on Systems, Man and Cybernetics, Part B, 34*(6), 2451-2465.

Carlos E. de M. Bicudo –Editorial Taxonomia - Instituto de Botânica/SMA. Disponível em http://www.biotaneotropica.org.br/v4n1/pt/editorial Ultimo acesso 21 de Agosto de 2010.

CASTLIELLO C; CASTELLANO G. e FANELLI A.; Meta-Data:Characterization of Input Features for Meta-Learning. ModelingDecisions for Artificial Intelligence, LNAI, 2005, 457-468

CHANDRASEKARAN, B; JOSEPHSON, J; BENJAMINS, V**.**What Are Ontologies, and Why Do We Need Them? In: IEEE Intelligent Systems. Vol. 1, 1999, p. 20-26.

CHAREST M.; DELISLE S. ;CERVANTES O. e SHEN Y.:Intelligent Data Mining Assistance via CBR and Ontologies, toappear in DEXA-PMKD Workshop. Poland, 2006

CHIKOFSKY, E. J.; CROSS, J. H. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, v. 7, n. 1, p. 13-17, 1990.

CIMIANO, P., Stumme, G., Hotho, A., & Tane, J. (2004). Conceptual knowledge processing with formal concept analysis and ontologies.In *Proceedings of the Second International Conference on Formal Concept Analysis (ICFCA 04)*.

CORCHO, O.; GÓMEZ-PÉREZ, A.A Roadmap to Ontology Specification Languages. In: EKAW00 – XII International Conference on Knowledge Engineering and Knowledge Management, 2000, p.80-96.

CORCHO, O., Fernández-López, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies: where is their meeting point? *Data & Knowledge Engineering 46*(1), 41-64. Amsterdam: Elsevier Science Publishers B. V.

CORCHO O., Fernández L. M., Gómez P. A., Vicente O.; WebODE: an integrated workbench for ontology representation, reasoning and exchange. Facultad de Informática. Universidad Politécnica de Madrid Campus de Montegancedo, s/n. 28660 Boadilla Del Monte. Madrid. Spain

CyC Project. Whta is ic Cyc? 2005. Disponível em: HTTP://www.cyc.com/cyc/technology/whatiscyc_dir/whatsincyc. Acessado em 01 de outubro de 2010.

DAM+OIL .disponível em <DAM+OILhttp://www.daml.org/2001/03/daml+oil-index.html> . Acessado em 15/08/2010.

DAVIES, John; DUKE, Alistair; STONKUS, Audrius. OntoShare:Using Ontologies for Knowledge Sharing. In Proceedings of the WWW2002 Semantic Web workshop, 11th International WWW Conference WWW2002, Hawaii, USA, 2002, p. 12-21.

Dejing Dou , Paea LePendu, SAC’06 Symposium on Applied Computingroceedings of the 2006 ACM symposium on Applied computing

Dejing Dou , Paea LePendu. Discovering Executable Semantic Mappings Between, Springer Berlin / Heidelberg Ontologies, 2007

DING, Y.; FENSEL, D. Ontology Library Systems: The key for successful Ontology Reuse.The first Semantic Web Working Symposium (SWWS1), Stanford, USA, 2001.

DOME Project Repository, disponível em: http://dome.sourceforge.net/. Ultimo acesso em 10 Agosto de 2010.

DUINEVELD, A.J. et al. A Comparative Study of Ontological Engineering Tools, Amsterdam, 1999.

DUNG NGUYEN XUAN1,LADJEL BELLATRECHE,GUY PIERRA, A : Versioning Management Model for Ontology-Based Data Warehouses: 8th International Conference on Data Warehousing and Knowledge Discovery (DaWak '06)

DRAGUT AND R. LAWRENCE. Composing mappings between schemas using a reference ontology. In *Proceedings of International Conference on Ontologies, Databases and Application of Semantics (ODBASE)*, 2004.

DUNG NGUYEN XUAN1,LADJEL BELLATRECHE,GUY PIERRA, A : Versioning Management Model for Ontology-Based Data Warehouses: 8th International Conference on Data Warehousing and Knowledge Discovery (DaWak '06)

FACT ,The FaCT System. Ultima alteração 29/04/2003. Disponível em http://www.cs.man.ac.uk/~horrocks/FaCT. Acessado em 26/07/2010

FAYYAD, U. M., SHAPIRO,G. P.; SMYTH, P.; UTHURUSAMY, R.Advances in Knowledge Discovery and Data Mining, AAAI Press, California, 1996.

FARQUNHAR, A.; FIKES, R.; Rice, J.The Ontolingua Server: A Tool for Collaborative Ontology Construction.Knowledge Systems Laboratory, 1996.Disponível em <ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-96-26.ps>. Acessado em maio de 2008.

FELIPPO.Ariani Di. Desenvolvimento de uma estrutura conceitual (ontologia) para a área de Nanociência e Nanotecnologia http://www.geterm.ufscar.br/ariani/Rel_Final_Nano.pdf

FENSEL, D. et al. *OIL in a nutshell*. In: Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW), Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 2000.

Férnandez, M.; Gómez-Pérez, A; Corcho, O. Methodologies and Methods for Building Ontologies.In: Férnandez, M.; Gómez-Pérez, A; Corcho, O. Ontological Engineering. London: Springer, 2004. PP 107-153.

FERNÁNDEZ, M., A. GOMEZ-PÉREZ e JURISTO, N. (1997): METHONTOLOGY: From Ontological Art Towards Ontological Engineering, Workshop on Ontological Engineering, AAAI.

FIGUEIREDO, Antonio Dias de. Métodos de Inovação Científica II. Mestrado em Engenharia Informática, Universidade de Coimbra, 2002.

FOX, Mark S. Manual TOVE. Disponível em <http://www.eil.utoronto.ca/tove/comsen/intro11.html>. Acesado em maio de 2008.

FOX, M.S., BARBECEANU, M., GRÜNINGER, M. 1995An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour, Computers in Industry, Vol. 29, pp. 123-134.

FOX, M.S. 1992, The TOVE Project: A Common-sense Model of the Enterprise, Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Belli, F. and Radermacher, F.J. (Eds.), Lecture Notes in Artificial Intelligence # 604, Berlin: Springer-Verlag, pp. 25-34.

FREITAS, Karine,OntoEditor: Um editor para manipular ontologias na Web. Disponível em<http://www.dsc.ufcg.edu.br/~copin/pesquisa/bancodissertacoes/2003/KarineFreitas.pdf. Último acesso em 04-08-2010.

GILCHRIST, Alan. Thesauri, taxonomies and ontologies - an etymological note.Journal of Documentation, v. 59, n. 1, 2003, p. 7-18.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ, M.; DE VICENTE, A. 1996. *Towards a Method to Conceptualize Domain Ontologies.*Workshop on Ontological Engineering.ECAI'96. Budapest. Hungary. PP: 41-52.

GÓMEZ PÉREZ, A., & Manzano Macho, D., (Eds.) (2003). *Survey of ontology learning methods and techniques.*Deliverable 1.5 OntoWeb Project Documentation. Universidad Politécnica de Madrid. Acessado Agosto, 2008, from http://www.deri.at/fileadmin/documents/deliverables/Ontoweb/D1.5.pdf

GOTTGTROY, P., Kasabov, N., & MacDonell, S. (2003, December). An ontology engineering approach for knowledge discovery from data in evolving domains.In *Proceedings of Data Mining 2003 Data Mining IV*. Boston: WIT.

GOTTGTROY, P., MacDonell, S., Kasabov, N., & Jain, V. (2005).*Enhancing data analysis with Ontologiesand OLAP*. Paper presented at Data Mining 2005, Sixth International Conference on Data Mining, Text Mining and their Business Applications, Skiathos, Greece.

GROSOF, B.; HORROCKS, I. Description Logic Programs:Combining Logic Programs with Description Logic. Disponível em

<http://www.daml.org/listarchive/joint-committee/att-1213/01-dlp-wp-v7-brief.pdf>. Acessado em abril de 2008.

GRUBER, T. R.*A translation approach to portable ontologies*. In: Knowledge Acquisition, 1993.

GRUBER, T. What is an ontology.<http://www-ksl.stanford.edu/kst/

what-is-an-ontology.html>. Acessado em abril/2008.

GUARINO, N. Formal ontology in information systems. In: IOS Press, A, editor, FOIS'98, TRENTO, ITALY, p. 3–15, 06 1998.

GUARINO, N. Understanding, Building and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development". International Journal of Human and Computer Studies, 1997, p. 293-310.

GUARINO, N. Understanding, Building, And Using Ontologies. In: Knowledge Acquisition Workshop. Canadá, 1996.

GUARINO, N. Formal ontology, conceptual analysis and knowledge representation. International Journal of Human and Computer Studies, 43(5/6):625–640, 1995.

HAN, J.; KAMBER, M. Data Mining:Concepts and Techniques. Morgan Kaufmann, 2001.

HONAVAR, V., SILVESCU, A., REINOSO-CASTILLO, J., ANDOFF, C., DOBBS, D.: Ontologydriven Information Extraction and Knowledge Acquisition from Heterogeneous, Distributed Biological Data Sources. Proceedings of the IJCAI-2001

Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources. (2001)

HAND, D. J. Discrimination and Classification, Chichester, U.K.: John Wiley and Sons, 1981.

HAROLD, E. R. The XML Bible. IDG Books, 2 edition, 1999.

HENRIQUE, Leandro. OntoEditor Editor de Ontologias. Disponível em: http://www.lcad.icmc.usp.br/~rosane/app_projeto_leandro.pdf. Último acesso em 04-08-2010

HORROCKS, I. Using an Expressive Description Logic: FaCT or Fiction?In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98). Morgan Kaufmann Publishers, San Francisco, California, 1998, p. 636-647.

HORROCKS, I.; SATTLER, U.; TOBIES, S. Practical reasoning for description logics with functional restrictions, inverse and transitive roles, and role hierarchies.In Proceedings of the first workshop on Methods for Modalities (M4M-1), 1999.

HOTHO, A., Staab, S., & Stumme, G. (2003). Ontologies improve text document clustering. In *Proceedingsof the 3rd IEEE Conference on Data Mining*, Melbourne, FL, (pp.541-544).

HASTIE, T.; TIBSHIRANI, R.; BUJA, A. Flexible Discriminant Analysis by Optimal Scorring, Journal of American Statistical Association 89(428): 1255-1270, 1994.

HOZO: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship" :Kouji Kozaki, Yoshinobu Kitamura, Mitsuru Ikeda, and Riichiro Mizoguchi. Proc. of the 13th International Conference Knowledge Engineering and Knowledge Management(EKAW2002), pp.213-218, Siguenza, Spain, October 1-4, 2002.

IBM DB2 Data Warehouse Edition.Disponível em:

<http://www-306.ibm.com/software/data/db2/dwe/>. Acessado em julho de 2008.

JESS, Java Expert System Shell,http://herzberg.ca.sandia.gov/jess/

JOE: Java Ontology Editor Disponível em <http://www.cse.sc.edu/research/cit/demos/java/joe/>. Última atualização 21/07/1999. Acessado em 25/08/2010

JONES, D; BENCH-CAPON, T; VISSER, P. Methodologies for ontology development.in Proc. IT KNOWS Conference, XV IFIPWorld Computer Congress, Budapest, August., 1998.

INFERENCE, Ontologies and Datawarehouse: www.inferencenetwork.com. Acessado em julho de 2008.

Jyoti Kamal, Tara Borlawsky, Philip R.O. Payne. Development of an Ontology-Anchored Data Warehouse Meta-Model : AMIA 2007.

KARLSRUHE, U. Institut für angewandte informatik und formale beschreibungsverfahren.http://www.aifb.uni-karlsruhe.de/. Acessado em julho de 2008.

KMGEN.Algorithme Information Technology services.Disponível em http://www.algo.be/ref-projects.htm. Acessado em 21/04/2010.

KOZAKI, K.; SUNAGAWA, E.; KITAMURA, Y.; MIZOGUCHI R.: Distributed Construction of Ontologies Using Hozo. The Institute of Scientific and Industrial Research (ISIR), Osaka University.2007. Disponível em < http://www2007.org/workshops/paper_19.pdf>. Acessado em 08/09/2010.

LANGLEY, P. (2000). The computational support of scientific discovery.*International Journal of Human-Computer Studies,* 53, 393-410.

LANGLEY P. (2006). *Knowledge, data, and search in computational discovery*. Invited talk at International Workshop on feature selection for data mining: Interfacing machine learning and statistics, (FSDM) April 22, 2006, Bethesda, Maryland in conjunction with 2006 SIAM Conference on data mining (SDM).

LASSILA, O.; SWICK, R. Resource Description Framework (RDF): Model and Syntax Specification.Recommendation, World Wide Web Consortium, 1999. Disponível em <http://www,w3,org/TR/REC-rdf-syntax/>. Acessado em março de 2008.

LENAT, Doug. Manual CYC.Cycorp, 1995. Disponível em <http://www.cyc.com/tech.html>. Acessado em abril de 2008.

LENAT, D.B.; GUHA, R. V. Building large knowledge-based systems .Massachusetts: Addison-Wesley, 1990. Pg 372.

LOUZADA NETO, FRANCISCO; DINIZ, CARLOS ALBERTO RIBEIRO. Técnicas estatísticas em Data Mining.Monografias Del IMCA no 31. Lima-Peru: IMCA, 2002.

LOWE, D.; WEBB, A.R.Optimized Feature Extraction and the Bayes Decision in Feed-Forward Classifier Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 13: 355-364, 1991.

MAEDCHE, A; STAAB, S; STOJANOVIC, N; STUDER, R; SURE, Y. Semantic portal - the seal approach, 2001.

MCLACHLAN, G. Discriminant Analysis and Statistical Pattern Recognitions, New York: Wiley, 1992.

MOSTOWFI F.; FOTOUHI F. e ARISTAR A. Ontogloss: An Ontology-Based Annotation Tool. 2005. Department of Computer Science, Wayne State University, Detroit, Michigan. Disponivel em < http://emeld.org/workshop/2005/papers/mostowfi-paper.html>. Acessado em 22/07/2010.

NEON TOOLKIT. Disponível em <http://neon-toolkit.org/wiki/Main_Page> , ultima modificação 22/09/2010. Acessado em 20/10/2010.

NOY, N.; FERGERSON, R.; MUSEN, M.The knowledge model of Protege-2000: Combining interoperability and flexibility. 2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France, 2000, p. 17-32.

NOY, F. N.; GUINNESS, D. L. Ontology development 101: a guide to create your first ontology. 2001. Disponível em: <http://ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.doc>. Acessado em 03 outubro de 2010.

NOYNEF, MusenMA. *SMART: Automated support for ontology merging and alignment*. SMI Report Number: SMI-1999-0813.

ONTOSTUDIO. Disponível em <http://semanticweb.org/wiki/OntoStudio>. Acessado em 28/07/2010.

ONTOPRISE. http://www.ontoprise.de/de/en/home/products/ontostudio.html, 2010. OntoStudio. Acessado em 13/10/2010.

WHAT IS AN ONTOLOGY? *FAQ Ontology.org*. Disponível em <http://www.ontology.org/main/papers/faq.html>. Acessado em abril de 2008.

SURE, YORK & STUDER, RUDI. On-To-Knowledge Methodology -Final Version. University of Karlsruhe, Germany. September 2002. P. 33 e seguintes.

ORACLE DARWIN DATA MINING Software.Disponível em: <http://www.oracle.com/technology/documentation/darwin.html>. Acessado em julho de 2008.

OWL Web Ontology Language Guide: W3C Recommendation 10 February 2004. W3C (2004-02-10). Disponível em: < http://www.w3.org/TR/owl-features/> Acessado em 15/09/2010.

O'CONNOR M.; KNUBLAUCH H.; TU S.; GROSOL B.; DEAN M.; GROSSO W. e MUSEN M., Supporting Rule System Interroperability on the Semantic Web with SWRL, ISWC, LNCS 3729. Berlin, 2005

PAN, D., & Pan, Y. (2006, June 21-23). Using ontology repository to support data mining.In *Proceedingsof the Sixth World Congress on Intelligent Control and Automation*, Dalian, China, (pp.5947-5951).

PAN, D., & SHEN, J. Y. 2005.Ontology service-based architecture for continuous knowledge discovery.In *Proceedings of International Conference on Machine Learning and Cybernetics*, *4*, 2155-2160.IEEE Press.

PROTÉGÉ FRAMES, <http://protege.stanford.edu/overview/pf-screenshots.html>. Acessado em julho de 2008.

PROTÉGÉ TUTORIAL,<http://www.eci.ufmg.br/mba/onto/>. Acessado em julho de 2008.

QUINLAN, J. C4.5: Programs for Machine Learning, San Francisco: Morgan Kaufmann, 1993.

RDF, Resource DescriptionFramework. Disponível em :<http://www.w3.org/RDF/>. Acessado em 15/09/2010.

RENNOLLS, K. (2005). An intelligent framework (O-SS-E) For data mining, knowledge discovery and business intelligence. Keynote Paper. In *Proceeding 2nd International Workshop on Philosophies*

and Methodologies for Knowledge Discovery, PMKD’05, in the DEXA’05 Workshops (pp. 715- 719). IEEE Computer Society Press. ISBN 0-7695-2424-9.

SANTI, S. M. Ontologias – Abordagens de Construção e Aplicações.UFRGS, 2000.

SARKER R. AND NEWTON C. 2002 Genetic Algorithm for Solving Economic Lot Size Scheduling Problem, Computers & Industrial Engineering, 42(2-4), pp189-198. (2009 ISI Impact Factor 1.491)

SAS Entreprise Miner. Disponível em:

<http://www.sas.com/technologies/analytics/datamining/miner/>. Acessado em julho de 2008.

SEMANTIC TURKEY. Disponível em < http://semanticturkey.uniroma2.it/>. Acessado em 01/10/2010.

SILVERMAN, B. Density Estimation for Statistics and Data Analysis, New York: Chapman and Hall, 1986.

SINGH, S., Vajirkar, P., & Lee, Y. (2003). Context-based data mining using ontologies.In Song, I., Liddle, S. W., Ling, T. W., & Scheuermann, P. (Eds.), *Proceedings 22nd International Conference on Conceptual Modeling*. Lecture Notes in Computer Science (vol. 2813, pp. 405-418). Springer.

SMITH, Barry. *Ontology and Information Systems*.Draft, 2001. Disponível em <http://ontology.buffalo.edu/smith/articles/ontologies.htm>. Acessado em março de 2008.

SMITH, M. K., WELTY, C; MCGUINNESS, D. L. (2003) "OWL Web Ontology Language Guide", http://www.w3.org/TR/2003/CR-owl-guide-0030818. Acessado em junho de 2008.

SPYNS, P., Meersman, R., & Jarrar, M. (2002). Data modeling versus ontology engineering, *SIGMOD Record Special Issue on Semantic Web, Database Management and Information Systems*, 31.

SOUTIPRIYA DAS, EUGENE CHONG, GEORGE EADON, JAGANNATHAN SRINVASAN. : Supporting Ontology-base Semantic Matching in RDBMS : 30th VLDB Conference, 2004.

STEPHEN REED AND D. LENAT 2002."Mapping Ontologies into Cyc". In: *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*. Edmonton, Canada, July 2002.

STUDERR, BENJAMINSVR, FENSELD. *Knowledge Engineering: Principles and Methods*. IEEE Transactions on Data and Knowledge Engineering 25(1-2):161–197.

Sure Y, Erdmann M, AngeleJ, StaabS, StuderR, WenkeD. *OntoEdit: Collaborative Ontology Engineering for the Semantic Web*. In: HorrocksI, HendlerJ (eds) First.

SUGUMARAN V. e STOREY V.C. Ontologies for conceptual Modeling:their creation,use, and management. Data & Knowledge Engineering 42 251 –271, 2002.

SWARTOUT, B; PATIL, R.; KNIGHT, K.; RUSS, T. Toward Distributed Use of Large-Scale Ontologies. 1996. Disponível em: <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/Banff_96_final_2.html>. Acessado em 15 de agosto de 2010.

TASY, Software de gestão de saúde.

<http://www.wheb.com.br/pt_br/solucoes.asp?menu=2.>

TENNISON, J.; SHADBOLT, N.R.; APECKS: a tool to support living ontologies. Disponível em <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/tennison/>. Acessado em 23/08/2010

TOPBRAID COMPOSER. Disponível em <http://www.topquadrant.com/products/TB_Composer.html>. Acessado em 02/10/2010.

TRINKUNAS J, OLEGAS VASILECAS. : Building Ontologies from Relational Database using Reverse engineering Methods .International Conference on Computer Systems and Technologies - *CompSysTech'07*

USCHOLD, M.; GRUNINGER, M. Ontologies: Principles, Methods and Applications. In: Knowledge Engineering Review. Vol. 11, No. 2, 1996.

USCHOLD, M. et al.The Enterprise Ontology.The Knowledge Enginee-ring Review.Vol. 13, Special Issue on Putting Ontologies to Use, 1998, p. 31-89.

USCHOLD, M .; KING, M. Towards a Methodology for Building ontologies. 1995. Disponível em: http://citeseer.istpsu.edu/uschold95toward.html. Acessado em 20 de outubro de 2010

USCHOLD, M.CONVERTING AN INFORMAL ONTOLOGY INTO ONTOLINGUA: SOME EXPERIENCES, 1996.

VASCONCELOS, K. A.: OntoEditor: Um editor para manipular ontologias na Web, 2003. Tese de mestrado. Universidade Federal de Campina Grande.

VIEIRA, André A., TANAKA, Astério K., MOURA, Ana Maria de C. Ferramenta para Extração de Ontologias a Partir de Banco de Dados Relacionais.

Relatório Onto Editor. Disponível em http://www.lcad.icmc.usp.br/~rosane/Relatorio_OntoEditor.pdf. Último acesso em 04-08-2010.

VILLELA, M. L. B.; OLIVEIRA, A. P.; BRAGA, J. L. 2004. Modelagem Ontológica no Apoio à Modelagem Conceitual. Anais do XVIII Simpósio Brasileiro de Engenharia de Software – SBES 2004, Brasília –DF, páginas 241-256.

Web semântica: ontologias, lógica de descrição e inferência,

<http://www.inf.unisinos.br/~renata/laboratorio/publicacoes/webmedia-webs.pdf>, pág.14. Acessado em julho de 2008.

WEBODE, Ontology Engineering Platform, Disponível em <http://webode.dia.fi.upm.es/WebODEWeb/index.html>. Acessado em 14/08/2010

WERBOS, P. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Ph. D. diss., Harvard, August, 1974.

WITTEN, I.; FRANK, E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Waikato: Morgan Kaufmann Publishers, 2000. 416 p.

WEISS, S.I.; KULIKOWSKI, C. Computer Systems that Learn: Classification e Prediction Methods from Statistics, Neural Networks, Machine Learning, and Expert Systems, San Francisco, California, Morgan Kaufmann, 1991.

W3C, The World Wide Web Consortium (W3C). Extensible Markup Language (XML). http://www.w3.org/XML.

XML Technology – W3C. disponível em <http://www.w3.org/standards/xml/>. Acessado em 15/08/2010.