UM ASSISTENTE ESPECIALISTA PARA ESPECIFICAÇÃO DE REQUISITOS

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pela Sra. Cecilia Inés Sosa Arias e aprovada pela Comissão Julgadora.

Campinas, 9 de junho de 1992

Ariadne M. B. R. Carvalho

Profª. Drª. Ariadne M. B. Rizzoni Carvalho

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação

UM ASSISTENTE ESPECIALISTA PARA ESPECIFICAÇÃO DE REQUISITOS

Cecilia Inés Sosa Arias

Orientadora: Ariadne Maria Brito Rizzoni Carvalho

Departamento de Ciência da Computação
IMECC-UNICAMP
Caixa Postal 6065
13081 - Campinas, SP - Brasil

8 de maio de 1992

Dissertação apresentada ao Departamento de Ciência da Computação
como parte dos requisitos exigidos para a obtenção
do título de Mestre em Ciência da Computação.

Este trabalho contou com o apoio financeiro da CAPES, CNPq e FAPESP

A meus pais

A meus irmãos

A Ivonne

A Hermano

AGRADECIMENTOS

Expresso minha gratidão a Ariadne, pela confiança, apoio e amizade com as quais o nosso trabalho foi enriquecido. Muito obrigada pela sua orientação.

À Universidade Nacional de Salta que tornou possível a realização deste trabalho.

A meus queridos primos, Hernán e Fatima, pela amizade sincera e apoio contínuo que recebi nestes anos.

À Inés, pelos três anos compartilhados de uma bela amizade. A Chris, pelo seu sorriso matinal. A Silvinha por colocar sempre a sua quota de paciência. A Marquinhos, por ter conseguido que goste de Pink Floyd. A Valery e Carol por esses meses de muitas alegrias. A Silvia pelos bons momentos. A Rubén pela sua valiosa ajuda neste trabalho.

A meus grandes amigos, Virgínia, Thelma, Tilica, Pite, Jussara, Ana, Maria, Carol e José, obrigada pelo seu carinho e amizade.

Aos companheiros de estudo, Lincoln e Adauto.

Por último queria agradecer aos amigos do DCC, LCA, do grupo vulcan e do voley.

De modo particular, ao Brasil, pela oportunidade oferecida.

Conteúdo

Lista de Figuras

Sumário

A especificação de requisitos, a primeira fase do desenvolvimento de *software*, sempre foi um dos principais focos de atenção da engenharia de *software*. Na atualidade, várias técnicas e metodologias pretendem solucionar, de maneira mais amigável, as dificuldades que esta fase apresenta. Neste trabalho de tese, define-se a arquitetura de um assistente especialista em metodologias cuja função é auxiliar o projetista durante a fase de especificação. Este assistente possibilita a especificação dos requisitos através de respostas dadas pelo usuário, em linguagem natural, a um questionário proposto pelo próprio assistente. A partir dessas respostas um modelo semi-formal, baseado no conhecimento sobre o modelo entidade-relacionamento, é gerado.

Abstract

Requirements specification, the first phase of software development, has been always one of the main concerns of the software engineering. Nowadays, several techniques and methodologies try to solve, in a friendly manner, the difficulties encountered in this phase. In this thesis the architecture of an assistant, expert in methodologies, is defined. Its function is to help the designer during this phase. The assistant allows requirements specification through answers, given by the user in natural language, to a questionnaire guided by the assistant. From the answers, a semi-formal model, based on the Entity-Relationship Model, is generated.

Capítulo 1

Introdução

A evolução das tecnologias de hardware dos sistemas computacionais nas últimas duas décadas permitiu a produção de máquinas cada vez mais poderosas a custos mais acessíveis. A tecnologia de *software*, entretanto, não evoluiu com a mesma rapidez, ocasionando uma demanda maior de sistemas de *software*, que começaram a alcançar tamanhos e níveis de complexidade que os tornaram difíceis de compreender e alterar. Como conseqüência imediata, estes sistemas passaram a não preencher seus requisitos, tornando-se freqüentemente ineficientes, pouco documentados e não confiáveis. Este problema ficou conhecido como a crise do *software*. Devido ao fato do *software* ser um produto lógico e não físico, seus custos se concentram no desenvolvimento e não na produção. Assim, esta crise engloba problemas associados a como desenvolver e manter um volume crescente de *software*. Muitas pesquisas foram feitas como tentativas de superar esta crise, mas o problema da especificação dos requisitos de *software* e de desenvolvimento de sistemas ainda continua sendo um desafio para os pesquisadores.

1.1 Paradigmas do Processo de Desenvolvimento de Software

Soluções para a crise de *software* começaram a ser apresentadas a partir dos anos 70. Surgiram assim novos **paradigmas** para entender o processo de desenvolvimento. Thomas Kuhn [Tak 90], em um texto clássico denominado "A estrutura das revoluções científicas", procurou fixar um modelo sob o qual o processo evolutivo da Ciência ao longo da história pudesse ser compreendido. Assim, definiu o conceito de **paradigma** como o conjunto de crenças e expectativas que guiavam a descoberta científica em um determinado momento. Nesse contexto, e segundo a classificação proposta por Mendes e Aguiar [Men 88], podemos considerar três tipos de **paradigmas**, que dão uma visão geral dos pontos de investigação.

O primeiro **paradigma** é a concepção do desenvolvimento de *software* como um artesanato. Uma visão tão radical dificilmente será encontrada hoje nos meios industriais e acadêmicos.

Encarar o desenvolvimento de *software* como uma obra pessoal leva ao segundo **paradigma**: a matemática como modelo desse desenvolvimento. Este paradigma se baseia na idéia de que resolver problemas é uma atividade básica dos matemáticos. Nesta mesma linha, vemos o surgimento de métodos formais que auxiliam durante o ciclo de vida do *software*.

O terceiro **paradigma** consiste em ver o desenvolvimento de *software* como engenharia. Segundo esta visão, a pesquisa é orientada à busca de métodos e técnicas que aproximem o mais

possível este processo de desenvolvimento das características de produtos tradicionais de engenharia.

No entanto, vemos que o processo de desenvolvimento de *software* é, em muitos aspectos, uma atividade mais próxima da realização de uma prova matemática do que da construção de um edifício. Entretanto, considerando-se que os produtos de *software* são avaliados por sua utilidade na solução de um problema, percebe-se que necessitamos de uma atitude basicamente de engenharia, além do conhecimento de matemática e do esforço individual do projetista.

Portanto, concordamos com Mendes e Aguiar que o processo de desenvolvimento de *software* é um processo híbrido que deve considerar todos estes três aspectos.

1.2 O Modelo Tradicional

Dentro do **paradigma** da engenharia, podemos citar o modelo de ciclo de vida do *software* (modelo em fases), que desde 1973 [Luc 87] tem sido utilizado no planejamento de produtos de *software*.

Numa primeira aproximação, este modelo pode ser resumido em cinco fases: especificação, projeto (*design*), implementação, teste e manutenção. Este é comumente conhecido como modelo em cascata devido ao fato de que os produtos vão passando de um nível a outro em contínua progressão.

A fase de especificação é normalmente associada à definição dos requisitos do sistema e a um primeiro planejamento da solução. A fase de projeto é composta essencialmente de duas partes: o projeto da arquitetura, onde são identificados os componentes do *software*, seus relacionamentos e estruturas e o projeto detalhado, que se encarrega da definição dos módulos e algoritmos.

A fase de implementação se associa à tradução para código de máquina e os testes cobrem os aspectos da integração do *software* e de sua aceitação. Finalmente, a fase de manutenção inclui as extensões, adaptações a novos ambientes e correções de erros de *software*.

Algumas limitações do modelo clássico podem ser consideradas [Nis 86]:

- O processo de desenvolvimento não é linear;
- As decisões de projeto são perdidas;
- Não fica claro como são feitas as transformações entre os vários níveis de projeto;
- A especificação, sendo informal, é responsável pela introdução de erros de ambigüidade que só são descobertos muito mais tarde, com altos custos de correção.

De maneira geral, o modelo clássico pode ser considerado válido numa situação onde é possível obter um conjunto razoavelmente completo de especificações para o *software* ainda no começo do ciclo.

1.3 Revisão do Processo de Desenvolvimento de Software

Uma nova visão do modelo de ciclo de vida do *software* surgiu com a concepção do atual objetivo da Engenharia de *Software* (ES), onde o usuário indica a especificação de um sistema a um ambiente a fim de obter dele a implementação do mesmo.

Assim vemos um avanço no sentido da formalização e mecanização de cada uma das fases do modelo do ciclo de vida. Esta nova concepção enfatiza as fontes de requisição do produto, os pontos de decisão projetista-usuário e o uso de protótipos para o aperfeiçoamento interativo do produto.

O modelo proposto na Figura 1.1 e que foi adotado como fundamento para o desenvolvimento deste trabalho, caracteriza-se pelo paradigma de programação automática e corresponde ao modelo de Balzer [Bal 85]. Nesta figura é possível perceber que a partir dos conceitos informais vamos obtendo os requisitos do sistema até conseguirmos uma especificação formal, que funciona como um protótipo sobre o qual é feita a validação.

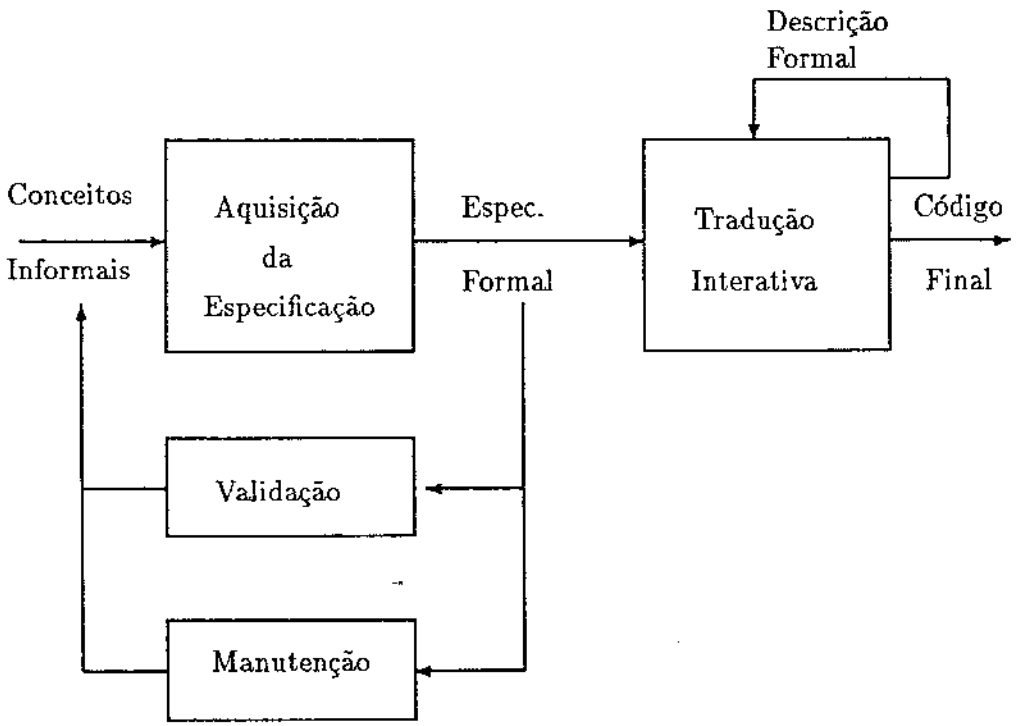


Figura 1.1: O Modelo para o Desenvolvimento de Software Proposto por Balzer

Protótipo é um artefato ou modelo de *software*. Seu objetivo principal é servir como instrumento de diálogo entre o projetista e o usuário, permitindo o aprimoramento das idéias através da sua utilização.

Este protótipo, por sua vez, serve como entrada para a fase do desenvolvimento formal, que é a geração mecânica da implementação, por meio de sucessivas traduções, até o código final do programa. Podemos identificar neste modelo a questão da aquisição da especificação como o problema central do desenvolvimento de *software*, justificando-se a necessidade de entendê-la para o aprimoramento do paradigma da programação automática.

1.4 A fase de Especificação no Ciclo de Vida do Software

A primeira idéia que surge quando se pensa em obter uma especificação é a de que ela representa uma "forma" para algo que se encontra em "pensamento".

Assim, uma especificação formal caracteriza um modelo que corresponde a uma abstração do objeto real sendo especificado, onde apenas detalhes relevantes são considerados [Cas 88]. Em um sentido mais rigoroso, uma especificação "formal" é feita através de uma linguagem ou notação gráfica que possua símbolos com sintaxe e semântica bem definidas, sem ambigüidade.

Resumindo, dentro deste paradigma concebe-se o processo de obtenção da especificação como a passagem de idéias ou conceitos expressos de maneira não-formal para uma representação formal, que expressa sem ambigüidades os requisitos do sistema.

Graficamente, na Figura 1.2 podemos ver o processo de desenvolvimento, que segundo a concepção de Lehman [Leh 84], é feito em duas etapas:

- **Processo de Abstração**: identificado como a obtenção de uma especificação formal com sintaxe e semântica precisas, através de transformações sucessivas;
- **Processo de Concretização**: transforma a especificação abstrata em um programa que implementa o sistema.

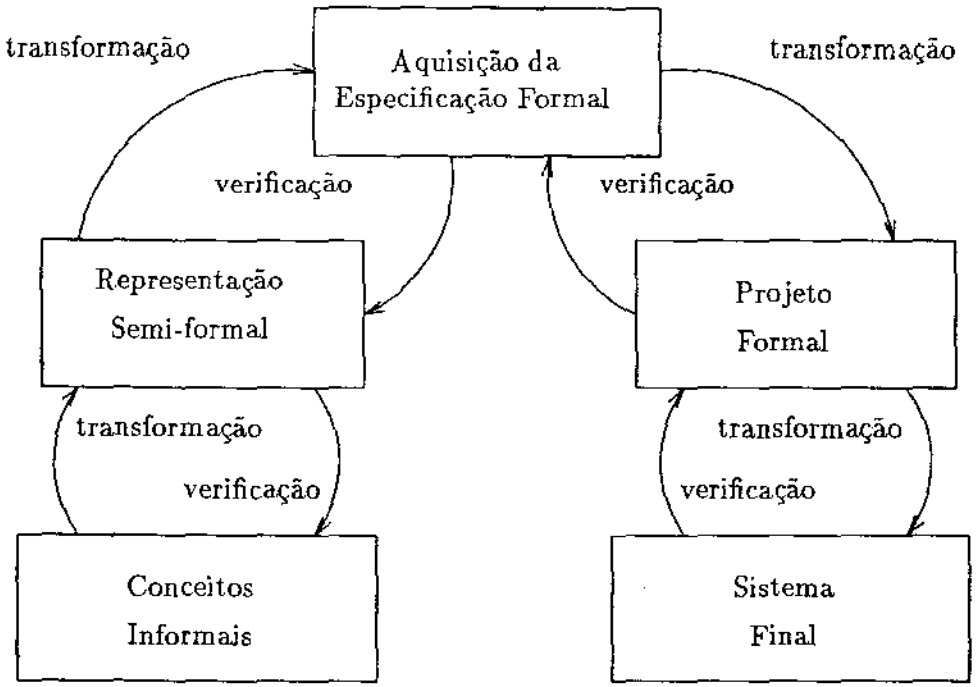


Figura 1.2: Decomposição do Processo de Desenvolvimento de Software

O início do processo de desenvolvimento consiste na exposição informal dos requisitos do sistema a ser desenvolvido. A partir dessa primeira verbalização, são derivadas por transformações duas

decomposições: uma primeira, que é a especificação formal do sistema e uma segunda, que é o programa que a implementa. Assim, surge a necessidade de se compreender com mais detalhes o que acontece quando se está obtendo uma especificação de requisitos.

Docker [Doc 89] destaca que as especificações são comumente imprecisas e incompletas, e algumas vezes inconsistentes, dando lugar ao surgimento de uma série de erros, devidos a:

- ambigüidades;
- inconsistências;
- má concepção das reais necessidades do usuário.

Na atualidade, técnicas de Inteligência Artificial (IA) estão sendo aplicadas como uma tentativa de solucionar esses problemas. Uma das aplicações mais imediatas de técnicas da IA aos ambientes de desenvolvimento de *software* é na construção de assistentes especialistas [Luc 87]. Assistentes especialistas têm por objetivo dar apoio às atividades do engenheiro de *software* em qualquer uma das fases do ciclo de vida. Assim, os assistentes servem como guias no processo, ajudando o engenheiro de *software* através de seu conhecimento especializado.

1.5 Assistentes Especialistas em Metodologias

Quando não é possível construir uma ferramenta para executar uma tarefa de maneira completamente automática, às vezes é possível construir uma ferramenta que possa assistir a um especialista na tarefa. Esta é a abordagem do assistente que, além de ser útil na prática, pode fazer contribuições importantes ao processo de construção de um sistema totalmente automático [Luc 87]. Um pressuposto básico na abordagem do assistente é a suposição de que o assistente é significativamente menos conhecedor que o especialista. Contudo, o objetivo de um assistente especialista é que ele "assista" e coopere com o usuário aceitando e raciocinando sobre a informação, propondo ações e respondendo a perguntas [Mor 86]. A importância da habilidade do assistente reside na possibilidade de sugerir um próximo passo lógico no desenvolvimento da área a qual ele "assiste". Alguns assistentes podem ser desenvolvidos segundo uma filosofia dual, ou seja, dando ao usuário a possibilidade de dirigir a consulta, ou deixar que o assistente o guie fazendo às vezes de tutores no uso de ambientes complexos de *software* [Pra 90].

Dois enfoques principais existem entre os assistentes especialistas. O primeiro enfoque refere-se ao assistente de programação. O assistente pode ser visto como "olhando" por cima do ombro do programador [Sch 88]. A filosofia deste assistente consiste em observar o que faz um especialista quando realiza sua tarefa. Nesta linha temos o Aprendiz de Programador (*Programmer's Apprentice*) [Wat 85], que é um assistente para programadores. O objetivo do aprendiz de programador é que este atue como um parceiro junior e crítico, tendo a ver com os detalhes e assistindo nas partes fáceis do processo de programação, deixando que o programador concentre-se nas partes difíceis do projeto. O segundo enfoque, no qual estamos interessados, são os assistentes especialistas em metodologias, cuja filosofia consiste em observar como os peritos em metodologias atuam na aplicação das mesmas.

Os assistentes especialistas em metodologias podem ser definidos como:

> **Ferramentas que utilizam explicitamente o conhecimento embutido numa metodologia para obrigar ou compelir à utilização das regras e diretrizes que constituem a metodologia. O planejamento e controle da solução, junto com a orientação da aplicação do método, devem ser apresentados ao usuário numa forma amigável e transparente, a fim de que este aprenda mais sobre a metodologia e interaja com o assistente, buscando a validação e aperfeiçoamento**

A partir dessa definição é possível estabelecer as características dos assistentes especialistas em metodologias

- **São ferramentas baseadas em conhecimento;**
- **Servem para encorajar ou mesmo compelir a utilização da metodologia;**
- **Proporcionam uma interface amigável com o usuário.**

A metodologia é expressa através de regras que definem seus princípios e regras que restringem práticas contrárias ao enfoque dado.

1.6 Motivação e Objetivos do Trabalho de Tese

A fase de obtenção da especificação continua sendo um motivo de preocupação e ponto crítico do ciclo de vida do *software*.

Assim, motivados pelos problemas apresentados nesta etapa, e que se relacionam à busca, coleta, entendimento e identificação das características ou propriedades relevantes à aplicação, decidimos construir uma ferramenta, denominada Ferramenta para Especificação de Requisitos - FER, que auxilie nas atividades relacionadas à fase de especificação de requisitos [Ari 92d]. Assim, pretende-se que os conceitos informais da aplicação a ser desenvolvida, que se encontram muitas vezes na cabeça do próprio usuário de forma não estruturada, sejam identificados e organizados em uma representação que possa ser melhor compreendida.

Este trabalho de tese concentra-se na definição da arquitetura de um assistente especialista em metodologias e na implementação de um protótipo do mesmo. A ferramenta fornece um ambiente que recebe como entrada os conceitos informais da especificação através de um diálogo entre o usuário e a ferramenta. Este diálogo, efetuado em um sub-conjunto da linguagem natural (LN), está fundamentado numa série de perguntas que serão apresentadas no capitulo III.

Uma vez obtidos os conceitos informais da aplicação, estes são transformados de acordo com o modelo entidade-relacionamento (MER) [Che 76], com o objetivo de estruturar as informações obtidas e de apresentá-las ao usuário num esquema sob a visão do MER.

1.7 Organização da Tese

Neste capítulo introdutório procuramos expor de maneira genérica a motivação para a construção do assistente, e o contexto onde este deve ser inserido. O próximo capítulo contém a descrição das ferramentas mais diretamente relacionadas à ferramenta proposta, enfatizando-se a sua arquitetura. No capítulo 3 é apresentado um modelo para aquisição da especificação que será estruturada

de acordo com o modelo entidade-relacionamento apresentado no capítulo 4. Todas as funções desempenhadas pelo assistente e as principais características de sua interface são implementadas em um protótipo de assistente descrito no capítulo 5. No capítulo 6 encontra-se um exemplo de utilização do assistente na aquisição da especificação de um sistema de informação de uma firma industrial. As conclusões e perspectivas de continuação deste trabalho são descritas no capítulo 7. No Apêndice A é apresentada a base de conhecimentos utilizada e no apêndice B é mostrado outro exemplo de utilização da ferramenta, na aquisição da especificação de um consultório médico.

Capítulo 2

Revisão Bibliográfica

Na literatura sobre Engenharia de *Software* pode-se encontrar vários estudos sobre técnicas destinadas a auxiliar no processo de aquisição da especificação. Da mesma maneira, existe uma grande quantidade de ferramentas automatizadas que, incorporando uma ou mais técnicas de Inteligência Artificial, contribuem para a etapa de especificação de *software*.

O objetivo deste capítulo é fazer um levantamento bibliográfico dessas ferramentas, levando-se em consideração os seguintes aspectos:

- O tema central deste trabalho é o desenvolvimento de um assistente especialista em metodologias;
- A importância da utilização de técnicas da análise de sistemas para a obtenção dos requisitos de *software*;
- A utilização do MER como método de especificação;
- A utilização de interfaces mais amigáveis visando facilitar a descrição das especificações.

Ao final deste capítulo é feita uma avaliação das ferramentas apresentadas mostrando-se, através de uma tabela, as suas características principais.

2.1 Ferramentas

2.1.1 TSER

TSER [Hsu 88] é uma ferramenta para especificação e modelagem de um banco de dados através da metodologia Entidade Relacionamento (ER) em dois estágios. O modelo ER foi desenvolvido para integrar tarefas de análise de sistemas com o projeto da base de dados e consiste na conceitualização da base de dados através de dois níveis (dois estágios), a saber:

- Nível de abstração;
- Nível operacional.

No nível de abstração são definidas as entidades e os relacionamentos (operações de uso e interação entre entidades). Neste nível, a conceitualização começa com um estágio de análise do

sistema, identificando os requisitos de informação da aplicação sendo analisada. Uma vez finalizada esta etapa é determinado o primeiro modelo entidade-relacionamento (chamado de modelo semântico SER), composto de:

- Entidades (SE);
- Relacionamentos (SR);
- Atributos;
- Plano de aplicação: hierarquia de sub-modelos SERs;
- Plano de classificação: generalização ou especialização de entidades.

Uma separação rigorosa entre as entidades e os relacionamentos (conhecimento operacional) é necessária no nível inferior da hierarquia, que é chamado de versão técnica do SER. Uma vez estabelecida a versão técnica é possível gerar um dicionário de alto nível para os usuários. O SER técnico é mapeado para outro modelo, chamado de modelo operacional entidade-relacionamento (OER). Cada objeto no modelo OER possui uma estrutura de dados pelo menos na terceira forma normal. Este é chamado de nível operacional.

O sistema TSER consta das seguintes partes:

- Construção do modelo;
- Algoritmos de mapeamento;
- Base de conhecimentos.

O usuário interage com TSER através do módulo de construção do modelo. Algoritmos de mapeamento são utilizados para transformar o modelo SER em um modelo OER. A base de conhecimentos é usada tanto internamente, pelos algoritmos de TSER, como externamente, pelos usuários. Através da metodologia de ER em dois estágios a ferramenta pode ser empregada como um assistente na modelagem de uma base de dados baseada em regras [Hsu 88].

2.1.2 Aprendiz de Requisitos

O Aprendiz de Requisitos (AR) [Reu 91] assiste um analista de sistemas na criação e modificação de requisitos de *software*. AR foi desenvolvido no contexto do projeto *Programmer's Apprentice* e é composto de três tipos de sistemas:

- Sistema de entrada: recebe as especificações e notifica ao analista as condições e inconsistências detectadas durante a entrada da informação;
- Base de conhecimento de requisitos: contém o conhecimento do AR sobre um domínio específico;
- Documento de requisitos: o sistema cria um documento de requisitos com o conteúdo da base de conhecimentos.

O AR não interage diretamente com o usuário final; ele assiste o analista de sistemas, sendo este último responsável pela comunicação com o usuário. O analista deve proporcionar as informações (requisitos) ao aprendiz de requisitos que, por sua vez, tem a tarefa de eliminar a ambigüidade das palavras introduzidas pelo analista. O problema da ambigüidade é solucionado através da biblioteca de *clichés* que possibilita inferir grandes quantidades de informação a partir dos enunciados do analista. O termo *clichés* aqui se refere a um método padrão de executar uma tarefa. AR utiliza a noção de *clichés* para representar, organizar e aplicar o conhecimento específico do domínio. A codificação do conhecimento em *clichés* é uma das partes mais importantes do trabalho e inclui tanto *clichés* gerais como específicos.

O AR verifica continuamente a consistência dos requisitos (buscando contradições) e completude e refina os enunciados através da classificação e de heurística. A classificação pode ser vista como um algoritmo de reconhecimento que aplica a informação contida na biblioteca de *clichés* à informação na base de conhecimentos. Quando uma palavra é associada a um dado *cliché*, o AR começa mostrando as pré-condições do *cliché* para ver se são contraditas ou satisfeitas (parcial ou totalmente).

Nesse mesmo contexto temos o SPECIFIER [Mir 91], um assistente inteligente que deriva especificações formais a partir de uma definição informal do problema, expressa em um sub-conjunto restrito da linguagem natural.

Um pré-processador analisa essa especificação informal e extrai os conceitos mais importantes. Associada a cada conceito, existe uma estrutura em forma de *frame*, chamada *template* que contém informação semântica sobre os conceitos. O pré-processador obtém os *templates* da base de conhecimentos, preenche os seus *slots* usando a especificação informal e "monta" os *templates* resultantes para obter uma representação interna da especificação informal. Essa especificação é depois usada para obter a especificação formal do problema. Essa ferramenta não consta da tabela de avaliação pois ela se concentra nos métodos de derivação da especificação formal. A obtenção da especificação informal não é o objetivo principal do trabalho e, portanto, sua descrição é bastante sucinta.

2.1.3 Uma Ferramenta Inteligente para o Desenvolvimento de Programas segundo o Método de Jackson

A ferramenta desenvolvida por Soares ([Luc 87];[Pes 86]) e que denominaremos AMJ para efeito de simplificação, é um assistente especialista em metodologias que auxilia no processo de programação através do método de Jackson [Jac 83]. O objetivo do assistente é obter um diagrama de Jackson que corresponda à estrutura de controle do programa indicado pelo usuário. Para isto, o usuário fornece a visão inicial dos diagramas de entrada e saída. O diagrama de entrada passa por um processo de verificação sintática e semântica e, se as entradas forem válidas, o assistente procura sintetizar um novo diagrama de entrada com uma estrutura "isomorfa" ao diagrama do programa desejado. Uma vez gerado o diagrama, o assistente passa à fase de verificação para determinar se ele é consistente com a especificação da entrada original. Os diagramas fornecidos pelo usuário são árvores com nós do tipo seqüência, iteração e seleção. O processo de completar e fundir os diagramas de entrada e saída é guiado por várias regras, como por exemplo:

> Fundir duas árvores, nível por nível, começando-se pelo nó raiz
> e prosseguindo no sentido da amplitude da esquerda para direita.

Ao efetuar as verificações finais o assistente verifica se o diagrama resultante contém o(s) diagrama(s) de entrada original(is) (diretamente ou de forma equivalente). Também verifica se os

elementos obrigatórios (elementos que sempre ocorrem no arquivo de entrada, como por exemplo *flags* de terminação) ocorrem no diagrama resultante.

O sistema AMJ é formado por um conjunto de regras que se dividem em três grupos:

- Regras de validação das informações de entrada;
- Regras para gerar diagrama de programa;
- Regras para testar o diagrama gerado.

Por último, a base de conhecimentos incorpora conhecimento sintático e heurístico sobre a metodologia e permite que o assistente resolva problemas típicos de processamento seqüencial de arquivos, tais como contagem de lote, quebras múltiplas e *balance line*.

2.1.4 Metodologia de Projeto de Banco de Dados Form-Driven Baseada em Conhecimento

Este trabalho [Man 87] define uma metodologia e um sistema especialista para projeto de base de dados (que denominaremos SEBD para efeito de simplificação), baseado na descrição de formulários (ou formas). Formulários do tipo comercial são utilizados para mostrar as especificações de requisitos. O conteúdo (objetos) dos formulários é obtido através de um processo de aquisição de requisitos (aqui chamado de aquisição de conhecimento contextual). Este conhecimento é facilmente capturado pela análise de sentenças em linguagem natural, que devem mostrar os aspectos funcionais do objeto sendo especificado. Essas sentenças são da forma:

< sentença > <frase sujeito> <frase verbo > < frase objeto >
[< frase complemento do objeto >]

Objetos que estão presentes na maioria das sentenças, seja como sujeito ou objeto, são chamados de "objetos forma" e constituem o centro do formulário. O usuário deve definir os objetos antes da construção das sentenças. Um processo de validação é efetuado para assegurar que todos os objetos definidos apareçam pelo menos numa sentença. A análise da sentença requer um dicionário para reconhecer verbos, preposições, etc. Um diagrama entidade-relacionamento é gerado a partir da análise das sentenças. Este determina as entidades e relacionamentos; os atributos são incorporados posteriormente durante o processo chamado análise do formulário.

A arquitetura do sistema especialista SEBD é composta de:

- Sistema de definição do formulário: provê um meio-ambiente de edição para definir o *layout* do formulário, as propriedades associadas, e o conhecimento contextual.
- Sistema de análise do formulário: escolhe um formulário por vez para ser analisado, e deriva o diagrama correspondente. O diagrama assim definido é acrescido dos resultados da análise de um novo formulário. O processo continua até que não existam mais formulários.

A base de conhecimentos do sistema especialista possui os seguintes tipos de regras:

- Regras de classificação de palavras, como por exemplo:

  Um verbo transitivo ou frase verbal é reconhecido como um tipo relacionamento

- Regras de identificação de entidades: são aplicadas para identificar as entidades do diagrama final. Estas regras indicam os campos do formulário que podem ser candidatos a chaves da entidade;
- Regras na fase de identificação de relacionamentos: permitem que um relacionamento seja substituído por um ou mais relacionamentos;
- Regras de controle de integridade.

2.2 Conclusão

Apesar de cada ferramenta apresentar características particulares, no que diz respeito a funções e técnicas suportadas, elas possuem vários fatores em comum, que são mostrados na Tabela 1, podendo-se destacar:

- Utilização de interfaces mais amigáveis;
- Utilização de uma base de conhecimentos que guia a estruturação das informações;
- Geração de modelos textuais obtidos através do processo de estruturação.

	Interface	Base de Conhecimentos	Modelo Textual
TSER	Editor	Conh. da aplicação (hierarquia e generalização heurística)	Geração de um dicionário de dados
AR	Descrição das especificações	Conh. do domínio da aplicação	Geração de um documento de requisitos
AMJ	Descrição dos diagramas de E-S	Conh. do método de Jackson	Geração da descrição de um diagrama de Jackson
SEBD	Sentenças em LN	Conh. que guia o processo de análise da especificação	Geração da descrição de um diagrama Entidade-Relacionamento

Tabela 1: Avaliação das ferramentas

Desta forma justifica-se a conveniência de se desenvolver ferramentas com a arquitetura de nosso assistente especialista em metodologias, mostrada na Figura 2.1, e que consta das seguintes partes:

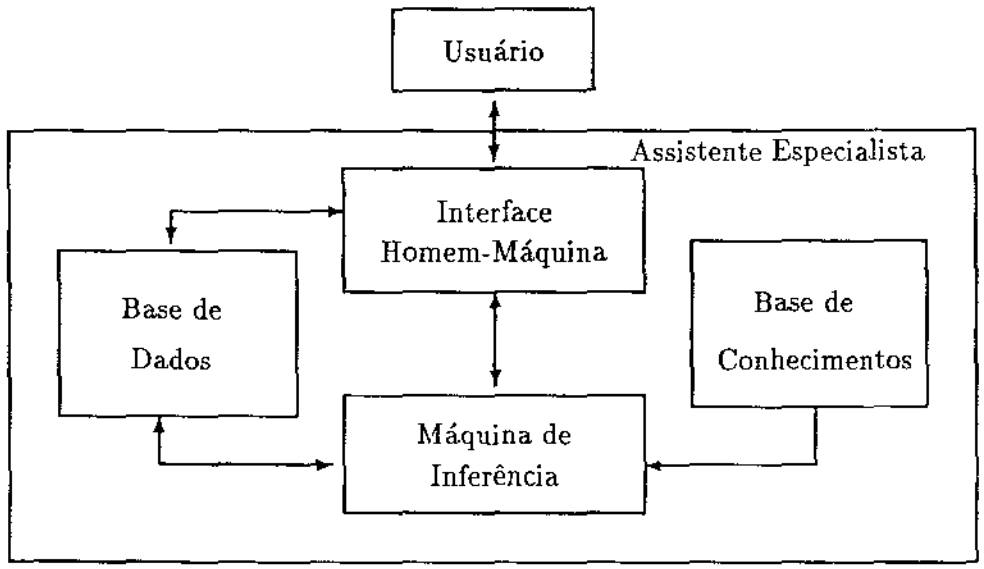


Figura 2.1: Arquitetura do Assistente Especialista em Metodologias

- Interface em linguagem natural: é um ambiente no qual o usuário fornece os conceitos informais da aplicação, através de um diálogo em um sub-conjunto da LN, guiado pelo assistente
- Base de conhecimentos: onde é armazenado o conhecimento sobre a metodologia de especificação (nesse caso o MER);
- Base de dados: onde é armazenado o conhecimento obtido através do diálogo com o usuário, e de onde se obterá, posteriormente, o modelo textual;
- Máquina de inferência: é a parte do assistente que consulta a base de conhecimentos para validar a especificação segundo o modelo entidade-relacionamento.

Capítulo 3

Aquisição da Especificação

O processo de abstração, como já havíamos mencionado no primeiro capítulo, representa a atividade de formular, clarificar e abstrair uma especificação formal. Assim, segundo Haeberer [Hae 88], a atividade de abstração pressupõe um complexo processo de:

- Seleção;
- Definição;
- Restrição;
- Generalização;
- Representação e
- Transformação do conceito da aplicação a ser desenvolvida.

Concluir todo este processo não é uma tarefa fácil. Na literatura, este processo é denominado análise, onde o objetivo não é efetivamente resolver o problema, mas determinar exatamente o que precisa ser feito para resolvê-lo [Wil 87].

Revisar a bibliografia relacionada à etapa da análise é uma tarefa trabalhosa, uma vez que são adotadas diferentes notações e abordagens para seu estudo. Assim, este capítulo é dedicado à discussão dos diferentes modelos de análise encontrados na literatura. Concluiremos apresentando os aspectos que foram selecionados para serem incluídos no modelo proposto e que são parte integrante da interface do assistente especialista.

3.1 Modelos de Abstração

No primeiro capítulo vimos que é de extrema importância a organização das especificações com vistas à generalidade, ou seja, o mecanismo básico da abstração deve permitir estruturar especificações complexas.

Neste contexto voltamos a falar de abstração no sentido das operações mentais que executamos para observar um domínio. Através da abstração, o indivíduo observa a realidade e dela abstrai entidades, ações, etc., consideradas essenciais a uma aplicação, excluindo todos os aspectos que julgar irrelevantes [Tak 90].

Como resultado dessa tarefa, obtém-se informações sobre a atividade a ser desenvolvida e pretende-se que a informação obtida seja:

- Objetiva: relate somente as informações necessárias à aplicação;
- Precisa: sem contradições e ambigüidades;
- Clara;
- Completa.

Com o objetivo de obter informações que cumpram essas condições foram desenvolvidos diferentes modelos de concepção da realidade e obtenção das informações, onde cada modelo enfatiza algum aspecto da situação a ser estudada. Assim, nas próximas seções apresentaremos vários modelos, cada um seguindo um prisma diferente.

3.1.1 Modelo Orientado a Objetos

Seguindo o paradigma de objetos, percebemos que a primeira grande fase no desenvolvimento de *software* consiste na análise do domínio da aplicação e na modelagem das entidades e fenômenos desse domínio que o projetista considerar relevantes para a aplicação. Esta fase é conhecida por modelagem conceitual e está condicionada a dois aspectos: abstração, que executamos para observar o domínio e captar sua estrutura em um modelo conceitual, e representação, que se refere às notações de representação adotadas [Tak 90].

Bruck [Bru 87] propôs um modelo que pretende obter do mundo da aplicação conceitos em termos de entidades e de operações e ações que relacionam essas entidades. De acordo com esse modelo, a fase de análise consta das seguintes fases:

1. Identificação das entidades e ações: o projetista analisa o domínio da aplicação e considera todas as pessoas, coisas e organizações que poderiam ser tomadas como entidades e todos os eventos que poderiam ser representados por ações, obtendo uma lista de entidades, uma lista de ações e uma lista de atributos das entidades;
2. Identificação dos objetos: os objetos são identificados através do relacionamento entre as ações e entidades que os realizam ou requisitam;
3. Identificação das entidades externas: análise das entidades que tenham relacionamentos com os objetos obtidos;
4. Representação dos objetos, suas operações (ações) e entidades externas;
5. Complementação da representação com os relacionamentos e fluxo de dados;
6. Comparação do modelo com os requisitos do sistema: análise do modelo obtido;
7. Definição das interfaces dos objetos;
8. Avaliação final : determinar a coesão do modelo como um todo.

Na utilização do modelo por parte do projetista, percebemos a necessidade de se conhecer o contexto dos objetos e a notação própria desta abordagem.

3.1.2 Modelo de Engenharia

Pressman [Pre 87] divide o ciclo de vida de um *software* segundo o prisma da engenharia em: planejamento, desenvolvimento, testes e manutenção. A fase de planejamento do ciclo de vida de um *software* é, por sua vez, um processo de definição, análise, especificação, estimativa e revisão. As três etapas associadas à fase de planejamento são:

1. Definição do sistema;
2. Planejamento de *software*;
3. Análise dos requisitos (especificações).

Durante a definição do sistema obtém-se uma visão completa da realidade da organização e do sistema a ser desenvolvido, alocando-se funções para cada elemento do sistema. Pressman define um *checklist* da análise de sistemas, que consiste em um catálogo da análise composta de um conjunto amplo de questões, com ênfase nas atividades associadas com a análise.

A especificação dos requisitos, por sua vez, é composta das seguintes etapas:

- Definição das metas e objetivos do *software*;
- Descrição da informação:
    1. Diagrama de fluxo de dados;
    2. Representação das estruturas de dados;
    3. Dicionário de dados;
    4. Descrição das interfaces do sistema;
    5. Interfaces internas.
- Descrição funcional:
    1. Funções;
    2. Narrativa do processamento;
    3. Restrições de processo.
- Critérios de validação:
    1. Limites de execução;
    2. Classes dos testes;
    3. Respostas emitidas pelo *software*;
    4. Considerações especiais.

3.1.3 Modelo de Sistemas de Produção

Nadler [Nad 71] estabeleceu uma metodologia que permite aplicar uma série de conceitos ao projeto de sistemas de produção, sem considerar a magnitude dos sistemas ou a índole da organização.

Esta metodologia se baseia na divisão da organização em níveis e na descrição de características sistêmicas para cada nível estabelecido.

Através da definição das características sistêmicas é possível descrever o sistema de produção. Suas características são:

1. Função: indica o que se deve obter;
2. Insumos (*inputs*);
3. Saídas (*outputs*): itens físicos ou serviços que expressam como se cumpre a função;
4. Seqüência: ordem de passos para converter *inputs* em *outputs*;
5. Ambiente: fatores físicos e sociológicos;
6. Catalizadores físicos: recursos físicos que se utilizam para converter os insumos em saídas ou resultados;
7. Agentes humanos: recursos humanos que servem como agentes dirigindo os catalizadores físicos para cumprirem uma função.

A definição das características sistêmicas para cada nível (ou sistema, segundo a notação de Nadler), permite estabelecer as inter-relações e compartilhamento dos recursos entre os níveis. Para isto, cada sistema recebe o *input* de níveis superiores e horizontais (itens físicos, humanos e informativos) e transmite o *output* para níveis inferiores ou horizontais (serviços e informação).

Entretanto, existem modelos que afirmam que o fluxo de informação constitui ponto fundamental da análise, já que a informação permite o início das atividades em uma organização.

3.1.4 Modelo de Sistemas de Informação

Senn [Sen 87], aponta duas estratégias amplamente utilizadas para determinar os requisitos de informação dentro de uma organização. Estas estratégias são:

- Fluxo de dados: onde é preciso conhecer
  - Quais os processos que integram o sistema;
  - Quais os dados utilizados em cada processo;
  - Quais os dados armazenados;
  - Quais os dados que entram e saem do sistema.

  A importância desta estratégia, centralizada nos dados, é devida ao fato de se considerar os dados como propulsores das atividades da organização;
- Estratégias de análise de decisão: completa a análise de fluxo de dados e realça o estudo dos objetivos e decisões a serem tomadas; é fundamental para determinar as necessidades de informação das pessoas que tomam as decisões.

Assim, com o objetivo de adquirir um conhecimento detalhado de todas as partes importantes da organização, Senn definiu um conjunto de 25 perguntas divididas nas seguintes áreas:

- Volume;
- Controle;
- Processos;
- Dados;
- Outras informações.

Entretanto, Hartman [Har 82] no seu livro "Manual de Sistemas de Información", detalha uma série de atividades a serem completadas a fim de determinar um sistema de informação dentro de uma organização.

As quatro etapas, definidas por Hartman, e que são de nosso interesse dentro do contexto de especificação de requisitos, são as seguintes:

- Estudo da organização existente;
- Identificação do fluxo de produtos;
- Identificação do fluxo de informação;
- Avaliação do sistema existente.

Cada uma das etapas é composta de questionários que guiam a análise de informação. O nível de detalhe das perguntas que fazem parte dos questionários é muito grande, o que leva a uma estrutura por demais complexa da situação da organização.

3.1.5 Modelo de Entrevistas

Loh [Loh 88] descreve um método de coleta de dados com o objetivo de obter informação sobre as entidades, objetos, relacionamentos entre entidades, hierarquias, atividades, funções e tarefas relevantes à organização. A coleta é feita através das respostas a um questionário contendo um conjunto de perguntas divididas em sete áreas, a saber:

1. A organização e seus objetivos;
2. As áreas funcionais da organização e seus objetivos;
3. Os Clientes/Usuários da organização;
4. Os funcionários e suas atribuições dentro de cada área funcional;
5. Os registros de cadastros, arquivos e formulários internos;
6. O ambiente externo relacionado à organização;
7. O futuro da organização.

Muitas das áreas de estudo propostas por Loh foram adotadas no modelo proposto e que será apresentado na próxima seção.

3.2 Modelo Proposto

Sabemos que não existe consenso entre os diferentes especialistas de como levar adiante a especificação de requisitos, mas vimos que uma das maneiras de obter informação tem sido as entrevistas de análise. Estas fazem parte do processo de especificação de requisitos e são consideradas de grande importância por determinar as bases do êxito ou fracasso no desenvolvimento de *software*. Assim, devido à utilidade das entrevistas e por serem estas uma forma mais "natural" de coleta de informação, decidimos utilizar um conjunto de perguntas que possibilitem um nível de abstração tão elevado quanto possível, e respostas abertas, onde os entrevistados possam dar um subconjunto amplo de respostas que lhes pareçam apropriadas [Ari 92b].

Dos modelos propostos por Senn, Loh e Hartman, foram incorporadas uma série de perguntas para fazer parte do modelo proposto. Do modelo de Pressman utilizamos a descrição funcional que permite uma melhor compreensão das estruturas funcionais dentro da organização.

As perguntas estão divididas em seis áreas a saber:

1. A organização, seus objetivos e ambiente externo relacionado a organização: esta área é fruto do consenso da maioria dos modelos e permite estabelecer as características da organização e seu ambiente externo;
2. Descrição funcional: permite refinar as funções da organização em detalhes;
3. Características estruturais da organização: permite completar as informações da organização e descrição funcional;
4. Os clientes/usuários da organização: permite conhecer as interfaces da organização;
5. Fluxo de informação: esta área está presente em todos os modelos e permite avaliar o fluxo de informação;
6. Futuro da organização: permite descobrir modificações futuras a serem introduzidas.

A seguir, mostraremos a lista de perguntas que fazem parte de cada área:

1. A organização.

   Questões

   (a) Indicar de que é composta a organização.
   (b) Indicar o objetivo da organização.
   (c) Indicar os fornecedores da organização e o que fornecem.
   (d) Citar outros orgãos externos à organização com os quais ela possui algum relacionamento.
   (e) Indicar que relatórios ou informações precisam ser passados para algum orgão externo.

   Resultados esperados

   - Obter uma declaração compreensiva dos objetivos do sistema;
   - Obter uma declaração geral do escopo onde se encontra a organização;

- Obter dados e funções relevantes à organização e seu meio ambiente.

2. Descrição funcional.

   Questões

   (a) Indicar a atividade básica da organização.
   (b) Com relação a atividade principal da organização, indicar os dados que são utilizados ou produzidos durante essa atividade.
   (c) Descrever informalmente como cada atividade (ou função) é realizada e que parte da organização a realiza.

   Resultados

   - Obter uma narrativa compreensiva das atividades do sistema e suas operações.

3. Características estruturais da organização.

   Questões

   (a) Indicar as partes que compõem a organização.
   (b) Indicar os objetivos de cada parte.
   (c) Se alguma parte da organização se relacionar com outra, indicar com quem e através de qual atividade.

   Resultados

   - Obter uma declaração compreensiva da estrutura da organização.

4. Clientes/Usuários da organização.

   Questões

   (a) Indicar quais os benefícios que os usuários recebem da organização.
   (b) Indicar com que funções interagem os usuários.
   (c) Dar sugestões em termos de atividade que os usuários gostariam de ter.

   Resultados

   - Uma análise interna das atitudes dos usuários e informação específica sobre os clientes.

5. Fluxo de informação.

   Questões

   (a) Para cada atividade fundamental descrever que dados utiliza, possui ou emite.
   (b) Indicar documentos, informes, ou outras saídas e a que setor (ou parte da organização) pertencem.
   (c) Indicar documentos, informes, que pertencem a algum orgão externo.
   (d) Propor uma atividade e indicar que informações ela requer para ser realizada.
   (e) Listar os novos elementos de dados que os usuários gostariam de ver no novo sistema. Indicar também de onde surgem e onde são utilizados.

Resultados

- Obter uma descrição precisa do fluxo de informação, incluindo documentos e outros suportes de dados que controlam as operações da organização [Har 88].

6. O futuro da organização.
   (a) Indicar, segundo o seu critério, que novas seções deveriam existir dentro da organização, especificando com quem deveriam se relacionar.
   (b) Indicar o que fazem as novas seções apresentadas acima.
   (c) Indicar que informações ou dados podem ser acrescentados a alguma atividade já descrita ou a uma nova atividade, para melhorar o seu funcionamento.
   (d) Indicar que novo serviço poderia ser prestado pela organização, dizendo que função executaria.

   Resultados
   - Apresentação das críticas e revisões internas.
   - Identificação das preocupações e questões a serem resolvidas.

Este conjunto de perguntas proporciona um meio para guiar o processo de obtenção da especificação que, na Figura 3.1, corresponde ao processo de "observação".

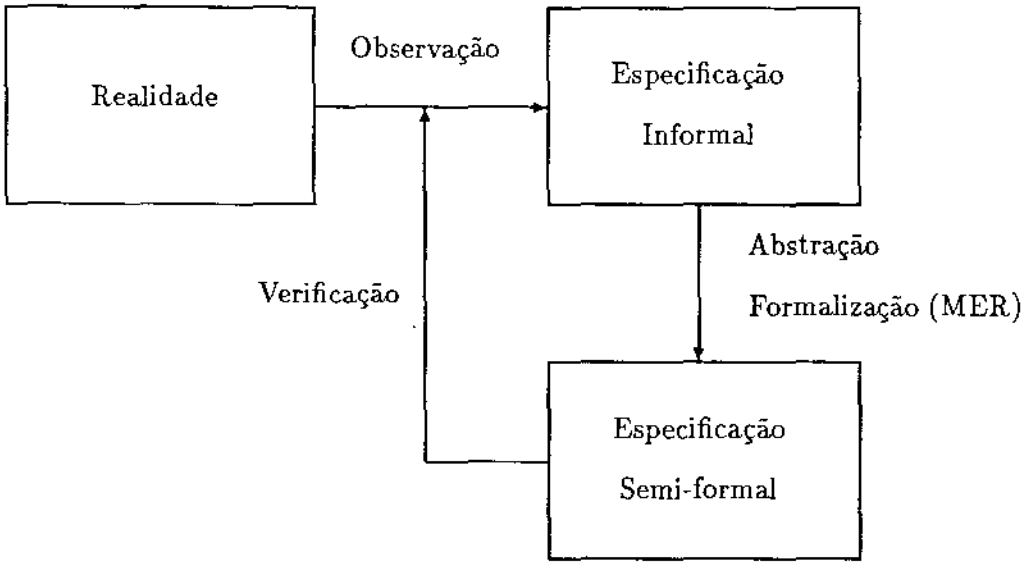


Figura 3.1: Processo de Obtenção da Especificação

3.3 Conclusão

Neste capítulo foi descrito um modelo de estruturação de uma especificação, através do uso de questionários, que possibilita:

- A obtenção de uma quase completa verbalização da especificação;
- Que o usuário tenha um melhor conhecimento sobre o problema;
- Iniciar a formalização das tarefas realizadas durante a fase de análise.

Veremos, no próximo capítulo, como transformar a especificação informal obtida em uma especificação semi-formal de acordo com as primitivas do MER.

Capítulo 4

Método de Representação da Aquisição: Modelo Entidade Relacionamento

Como já mencionado anteriormente este trabalho se concentra na etapa de abstração do ciclo de vida do *software*, até conseguirmos uma especificação dos requisitos da aplicação. Na prática, a especificação não existe como objeto atômico, mas sim como uma coleção estruturada de sub-especificações resultantes do processo de abstração. Neste processo vamos obtendo uma estrutura formada por especificações parciais EP e representações informais RI. Cada representação informal RI nada mais é que a verbalização V da aplicação (quando se trata da verbalização inicial), ou sub-aplicações (chamadas mais comumente de sub-sistemas) derivadas do sistema geral [Hae 88]. No processo de desenvolvimento cada representação informal $RI_j \in \{V_i\}_{i=1,n}$ é depois transformada em uma especificação parcial EP_j que, no nosso modelo, corresponde à especificação segundo a metodologia MER [Ari 92a].

Para conseguirmos transformar uma especificação informal RI em uma especificação parcial EP, precisamos definir os conceitos primitivos do modelo MER, que servirão como base para o processo de transformação.

Neste capítulo apresentaremos a semântica do MER e definiremos uma "meta-linguagem" que nos possibilitará descrever a especificação semi-formal da aplicação a ser desenvolvida.

4.1 Semântica do MER

A análise de dados utiliza-se do MER como modelo gráfico semi-formal para o entendimento e a visão dos dados do sistema e de seus relacionamentos, usando a representação resultante como um instrumento para o aprimoramento da aquisição da especificação [Nis 86]. O ponto positivo desta abordagem, e que contribui para sua divulgação, é a utilização de uma linguagem diagramática, bastante intuitiva, de representação. Um ponto negativo é o fato da abordagem permitir a representação diagramática de somente algumas propriedades estáticas, ficando a representação das demais para uma linguagem textual [Heu 88].

Contudo, o uso do MER, e de modelos de MER estendido para modelagem do esquema conceitual, é uma técnica bastante utilizada. Sabemos que os diferentes métodos de representação

trazem inerentes uma visão da realidade, que destaca alguns aspectos específicos do problema em detrimento de outros. No entanto, é interessante organizar a informação sobre a realidade em termos do modelo MER, já que este modelo capta e preserva alguns aspectos semânticos do mundo real [Che 83b].

A partir da definição da visão multinível proposta por Chen [Che 76], podemos observar que o primeiro nível consiste na identificação das informações de acordo com a concepção de entidade, relacionamento e atributos.

Assim, é possível identificar os conceitos primitivos básicos como sendo:

- Entidade;
- Relacionamento;
- Atributos.

4.1.1 Entidade

Segundo Lindgreen [Lin 87] entidades são concepções, ou seja "construções" de nossa mente que são convenientes quando interpretamos alguma coisa no Universo de Discurso (U.D.)[1]. Portanto, vemos que uma entidade pode ser entendida, de forma geral, como a abstração de um objeto do mundo real sobre o qual se deseja guardar informação, onde objetos do mundo real são os seres, fatos, as coisas e os organismos sociais.

4.1.2 Relacionamento

Um relacionamento representa um conjunto de conexões lógicas entre entidades. Segundo Kent [Ken 83] um relacionamento é a classe de conexão que inter-relaciona entidades num mesmo conceito ou fato e que possui um nome e um número finito de coisas que este pode conectar (grau). Se o grau é dois o relacionamento é chamado binário; neste caso pode-se definir um par ordenado (e_1, e_2) onde e_1 e e_2 são elementos ou instâncias das entidades E_1 e E_2, respectivamente. Seguindo esta notação, a Figura 4.1 mostra o relacionamento como um conjunto de pares ordenados (f, d), onde f pertence à entidade *Funcionários* e d pertence à *Departamentos* e tal que o par (f, d) representa o fato de f estar lotado em d. A representação diagramática do relacionamento, conforme vemos na Figura 4.1, é um losango.

Chen associa a cada conjunto de relacionamentos os papéis[2] que os elementos ou instâncias das entidades desempenham naquele relacionamento, ou seja, a função que uma entidade desempenha num relacionamento. Por exemplo, na Figura 4.1 teríamos os papéis "funcionário lotado em" para cada instância de funcionário e "departamento lota" para cada instância de departamento. Neste trabalho não usaremos o conceito de papéis, com exceção de casos de auto-relacionamentos, pois se o nome do relacionamento for bem escolhido os papéis ficam óbvios.

Os tipos de relacionamentos são:

- **Um-para-um**: quando uma instância de uma entidade interage com uma instância de outra entidade. R é da classe 1:1 sse $(a_k, b_m), (a_k, b_n), (a_r, b_m) \in R$; então $a_k = a_r$ e $b_m = b_n$

[1] Parte seleta ou hipotética do mundo real

[2] Em inglês: roles

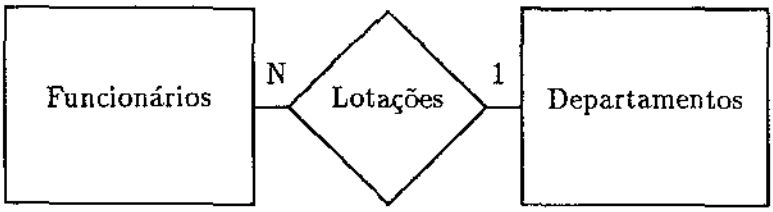


Figura 4.1: Exemplo de um Diagrama MER

- **Muitos-para-um**: quando uma instância de uma entidade interage com várias instâncias de outra entidade. Seja um relacionamento

$$R \subseteq \{(a_i, b_j), 1 \leq i \leq p, 1 \leq j \leq q\} .$$

R é da classe N:1 sse $(a_k, b_m) \in R$ e $(a_k, b_n) \in R$; então $b_m = b_n$.

Se um relacionamento é N:1 num sentido, ele é 1:N no outro sentido.

- **Muitos-para-muitos**: quando várias instâncias de uma entidade interagem com várias instâncias da outra entidade.

- **Auto-relacionamento**: Se R é um relacionamento que conecta instâncias da entidade E e instâncias da mesma entidade E, então R é denominado auto-relacionamento. Neste tipo especial de relacionamento nos vemos obrigados a fazer uso do conceito de "papéis". É comum usar as palavras "e","de" e "tem como" para aplicar aos papéis do auto-relacionamento, pois os auto-relacionamentos em geral modelam hierarquias ou subordinações.

- **Relacionamento triplo**:

  No caso de relacionamento triplo, como o apresentado na Figura 4.2, se o usuário não especifica um nome para o relacionamento, este pode ser definido com as iniciais dos nomes das entidades [Set 86]. Portanto, o nome criado foi M_P_R, onde M significa Materiais, P significa Pedidos e R significa Requisição.

4.1.3 Atributos

Atributos das Entidades

O processo de abstração que realizamos ao definir uma entidade, não será de grande valor se não associarmos a essa abstração as informações que desejamos guardar sobre os objetos. Assim, Chen [Che 76] considera estas informações como atributos da entidade, isto é, como funções que atribuem a um conjunto de entidades um conjunto de valores. Uma crítica usualmente feita ao MER é a falta de critérios para que o modelador decida se uma determinada correspondência entre entidades deve ser representada como um atributo de uma entidade ou como um relacionamento, como por exemplo, para decidir se cor é atributo da entidade automóvel, ou se existe um relacionamento coloração relacionando a entidade automóvel com a entidade cor [Heu 88]. A opção

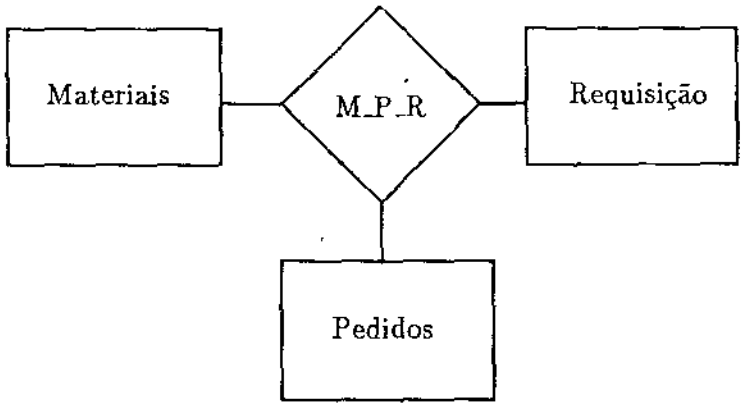


Figura 4.2: Relacionamento Triplo

atributo seria escolhida se dentre as propriedades dinâmicas[3] não existissem eventos cuja ocorrência resultasse em aparecimento/desaparecimento de cores mostrando não ser necessário considerar a entidade cor.

Atributos dos Relacionamentos

Atributos dos relacionamentos são, de acordo com Kent [Ken 83], "fatos sobre fatos" e podem ser de dois tipos:

- Composições: são aqueles atributos (ou fatos) compostos de atributos das entidades ou atributos compostos de vários relacionamentos, como por exemplo:

empregadores têm secretárias

secretárias tem número de telefone

mas o que nos interessa é que

empregadores tem número de telefone

ou seja

X é o número de telefone da secretária (sem especificá-la) de Y

- Restrições: quando o mesmo relacionamento é mantido separadamente para diferentes entidades, então é necessário "restringir" os casos. Exemplo:

empregadores são donos de estoque

departamentos são donos de projetos

[3]Descrição das alterações de estados que ocorrem sobre a base de dados

4.2 Transformação da Especificação em Primitivas do MER

Uma vez obtida a especificação informal através das respostas em linguagem natural, vemos a necessidade de estabelecer a correspondência entre os termos da linguagem e os conceitos primitivos do MER, a fim de obter a especificação semi-formal correspondente.

Portanto, começaremos por definir a unidade básica da "meta-linguagem" que servirá para descrever o conceito de entidade:

tipo(entidade,Nome).

Entende-se por essa definição que temos um tipo cuja classe é *entidade* e cujo nome é o conteúdo da variável *Nome*. Entretanto, para chegar a esse ponto, várias coisas precisam ser definidas e esclarecidas pois, como já havíamos dito anteriormente, as informações sobre as entidades, relacionamentos e atributos são proporcionadas pelo usuário, através de um subconjunto da linguagem natural. Então vemos que é preciso estabelecer uma correspondência entre as sentenças em linguagem natural e as primitivas do MER, de tal forma que quando o usuário responde à pergunta

Quais são os usuários da organização?

com

Os usuários são professores e funcionários

possamos obter uma relação[4] que diz

Professores e funcionários são entidades

definida através das seguintes cláusulas:

tipo(entidade,Professores) e tipo(entidade,Funcionários).

O fato da resposta ser analisada gramaticalmente nos leva a uma divisão da frase em categorias gramaticais (nomes, verbos, adjetivos, etc.) para as quais é possível expressar correspondências no modelo entidade-relacionamento. Diversos trabalhos ([Sco 89]; [Loh 88]; [Dol 88]; [Lin 83]) têm estabelecido esta correspondência entre elementos da linguagem natural e elementos do modelo conceitual, principalmente o trabalho de Chen [Che 83a]. Assim, possíveis entidades são identificadas através das categorias gramaticais **sujeito** e **objeto** [Loh 88]. Esta correspondência é resumida por Chen na seguinte regra:

Um nome comum (tal como "pessoa" ou "cadeira") corresponde a um tipo entidade no diagrama entidade-relacionamento

Portanto, numa frase como

O conhecimento torna a alma jovem[5]

tanto "conhecimento" como "alma" são do tipo entidade.

[4] Relação aqui é utilizada para definir a associação entre os conceitos primitivos "entidade" e "relacionamento"

[5] Leonardo da Vinci

A presença de um verbo é uma forte indicação de um relacionamento entre entidades [Set 86]. A regra de Chen correspondente a esta categoria gramatical diz que:

Um verbo transitivo corresponde a um relacionamento no diagrama entidade-relacionamento.

No exemplo anterior, "tornar" é um verbo transitivo e, portanto, corresponde a um relacionamento.

Então vemos que os **verbos** nos indicam a presença de um relacionamento que dará lugar ao surgimento da seguinte estrutura da "meta-linguagem"

tipo(relacionamento,Relacionamento)

onde um relacionamento entre duas entidades pode ser visto como dois tipos de relação. A primeira é uma relação de um determinado tipo entre uma entidade e um relacionamento (por exemplo, na Figura 4.1 a relação seria do tipo N entre *Funcionários* e *Lotações*). A segunda relação é uma relação de outro tipo entre a segunda entidade e o mesmo relacionamento (na Figura 4.1 a relação seria do tipo 1 entre o relacionamento *Lotações* e a entidade *Departamentos*). Essas relações dão lugar ao surgimento a seguinte estrutura da "meta-linguagem":

tipo(relação,Entidade,Tipo,Relacionamento).

O caso de restrições (relacionamentos com o mesmo nome) é solucionado através da seguinte estrutura da "meta-linguagem", que indica o relacionamento e as entidades conectadas:

tipo(par_ordenado,Entidade1,Relacionamento,Entidade2,Tipo1,Tipo2).

Os estudos realizados por Loh sugerem que, em verbos como **ter** ou **possuir**, o objeto é um atributo da entidade sujeito. Dolder [Dol 88] afirma que o conceito atributo é um relacionamento com cardinalidade 1:1, isto é, um relacionamento que possui essa propriedade e o conceito que denota essa propriedade. Adjetivos denotam a presença de atributos. No exemplo anterior vemos que "jovem" é atributo da entidade "alma". A regra diz:

Um adjetivo corresponde a um atributo da entidade no diagrama entidade-relacionamento
Um advérbio corresponde a um atributo do relacionamento no diagrama entidade-relacionamento
Em sentenças da forma **O X de Y é Z**, quando **Z** não é um nome próprio, podemos tratar **X** como um atributo de **Y**

Exemplo: A cor da xícara é azul celeste

"Azul celeste" não é nome próprio; então podemos inferir que "cor" é atributo da entidade "xícara". Com todo este conhecimento é possível distingüir os atributos tanto das entidades como dos relacionamentos e armazená-los na unidade básica da "meta-linguagem" que diz:

tipo(atributo,Atributo)
e
tipo(possui,Entidade,Atributo)
ou
tipo(possui,Relacionamento,Atributo).

Estas regras estão embutidas na gramática, que será descrita no próximo capítulo; assim, toda resposta é analisada gramaticalmente decidindo-se, a cada passo, que palavras são entidades, relacionamentos ou atributos. À medida que é feita a análise, as primitivas são armazenadas na base de dados, de onde serão posteriormente extraídas para serem revisadas, verificando-se sua consistência e completude.

4.3 Conclusão

Neste capítulo, vimos que para conseguir uma especificação semi-formal, através da descrição textual, vários passos precisam ser seguidos:

1. Obter do usuário as representações informais da aplicação na forma de sentenças gramaticais num subconjunto da língua portuguesa;
2. Transformar as representações informais obtidas em especificações parciais escritas de acordo com as primitivas da "meta-linguagem". Nesta transformação, para cada categoria gramatical obtemos uma primitiva da "meta-linguagem".
3. Gerar o modelo textual.

Entre as razões para a escolha do modelo MER podemos destacar:

- O modelo MER é um método amplamente difundido;
- Existe uma literatura considerável sobre as experiências de utilização do mesmo, o que possibilita obter conhecimento mais amplo sobre a sua aplicação;
- Possibilidade de criação de um modelo de dados que nos permita uma boa compreensão dos relacionamentos entre esses dados [Bor 87].

Capítulo 5

Assistente Especialista no Modelo Entidade-Relacionamento

Como vimos no segundo capítulo, o assistente especialista em metodologias, por nós desenvolvido, consta das seguintes partes [Ari 92c]:

- Base de Conhecimentos
- Base de Dados;
- Máquina de Inferência;
- Interface em Linguagem Natural.

Nas próximas seções, passaremos a descrever cada uma das partes da ferramenta, cuja arquitetura será mostrada novamente na Figura 5.1 por questão de conveniência.

Antes de cada descrição, entretanto, uma introdução teórica será apresentada. Na Seção 5.1, descreveremos os vários tipos de representação do conhecimento e finalizaremos com a representação do conhecimento por nós utilizada na construção do assistente. Na Seção 5.2, mostraremos a estrutura de armazenamento na base de dados. Na Seção 5.3, será indicada a estratégia de manipulação do conhecimento (máquina de inferência). Finalmente, na última seção, apresentaremos a interface da ferramenta, juntamente com um resumo das diferentes gramáticas existentes e com a definição da gramática utilizada pelo assistente.

5.1 Base de Conhecimentos

5.1.1 Representação do Conhecimento

Existem diferentes métodos para representação do conhecimento, cada um com as suas características de utilização e conceitualização do mundo real. Nesta seção apresentaremos uma revisão dos modelos de representação do conhecimento, sob o ponto de vista da abstração do conhecimento da realidade (chamado de nível de conhecimento [Car 86]), não nos preocupando com o tipo de acesso ou com o modo de implementação dos esquemas de representação.

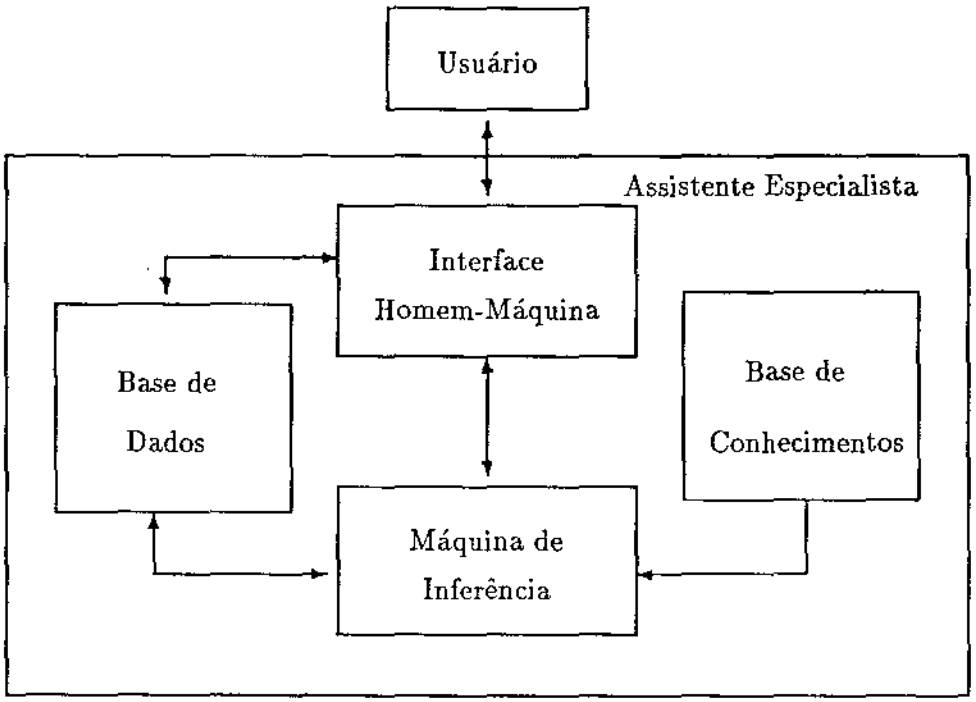


Figura 5.1: Arquitetura do Assistente Especialista em Metodologias

Os diferentes esquemas de representação surgiram de motivações concretas em campos específicos. Por exemplo, as redes semânticas surgiram dos estudos sobre o processamento de linguagem natural e sobre a teoria da estruturação da memória humana; os *frames* [Bar 81] surgiram da problemática de descrição dos cenários e situações concretas da vida real, a lógica surgiu da prova automática de teoremas, dos formalismos para análise de linguagem e dos sistemas de produção.

As vantagens e desvantagens de cada esquema são determinadas sem que se tenha uma teoria das linguagens de representação do conhecimento que possibilite uma análise num mesmo plano.

Nas próximas seções, analizaremos os diferentes formalismos para representação do conhecimento mais comumente citados ([Car 86]; [Car 88]; [Har 88]; [Bar 81]; [Mal 87]).

Redes Semânticas e Frames

Tanto nas redes semânticas quanto nos *frames*, o conhecimento é organizado em função dos objetos e eventos do universo de aplicação. Uma malha (rede semântica) representa a combinação de duas propriedades importantes. Por um lado, permite a descrição de taxonomias hierárquicas superclasse-subclasse e, por outro, possibilita a descrição de propriedades para cada classe (pares atributo-valor). Segundo King [Har 88], rede semântica é o esquema de representação mais geral onde, através da organização hierárquica induzida pelos arcos, é possível realizar inferências. Esta facilidade faz com que as redes semânticas resultem numa forma de representar os domínios com taxonomias complexas. No entanto, a interpretação dos arcos e nós pode ser ambígüa, levando a problemas de confiabilidade nas inferências realizadas. Nas redes semânticas, os arcos indicam o

predicado invocado e os seus argumentos. Esta notação possibilita uma clara correspondência com o cálculo de predicados.

Dentre as vantagens da estrutura de rede semântica temos:

- Flexibilidade: os nós são definidos à medida que precisamos deles;
- Hereditariedade: surge das relações do tipo é-um, sendo difícil o tratamento de exceções;
- Naturalidade na representação: representação baseada em lógica.

Frames são formalismos que podem ser vistos como uma generalização das redes semânticas [Mal 87], oferecendo mecanismos para tratar exceções e valores *default*. Estes mecanismos são necessários na representação de domínios da realidade e não são fáceis de implementar com lógica *standard*. A inclusão de procedimentos nos *frames* reúne, numa única estratégia de representação, duas formas complementares de afirmar alguma coisa sobre os fatos, assim como armazená-los:

- Representação declarativa: quando representamos um fato na forma declarativa, simplesmente asseguramos que o fato é verdadeiro;
- Representação procedimental: consiste num conjunto de instruções que, quando cumpridas, chegam a um resultado congruente com o fato.

Frames são um formalismo geral, potencialmente poderoso, que deixa muito nas mãos de quem o utiliza [Car 86].

Representação Baseada em Lógica

A lógica é a linguagem que permite as deduções ou inferências realizadas dentro do formalismo, afirmando ou não coisas que satisfaçam a realidade. Portanto, uma representação é simplesmente uma assertiva sobre o mundo, que pode ser verdadeira ou falsa. Dentre as vantagens essenciais da lógica temos:

- A sua capacidade de descrever qualquer domínio;
- A possibilidade de expressar conhecimento incompleto;
- Naturalidade na expressão do conhecimento sobre um problema;
- Precisão, flexibilidade e modularidade.

A vantagem principal dos formalismos baseados em lógica é a possibilidade de expressar uma grande variedade de situações mediante o uso complexo de conectivos e quantificadores, junto à combinação de símbolos de função. As cláusulas de Horn são restrições à lógica de primeira ordem que não têm sido aplicadas a domínios complexos, pelas dificuldades de controlar o crescimento desmesurado das inferências realizadas.

Representação Baseada em Regras

A representação baseada em regras é um tipo de representação do conhecimento que tem ampla expressividade e é bastante efetiva para representar associações empíricas obtidas como resultado de anos de experiência resolvendo problemas em um domínio específico. O conhecimento é estruturado como um conjunto de fatos, cada um referenciando um ou mais objetos.

Dentre as vantagens da representação baseada em regras [Bar 81] podemos citar:

- Modularidade: cada regra define uma pequena peça relativamente independente de conhecimento;
- Incrementabilidade: novas regras podem ser adicionadas à base de conhecimentos, de modo relativamente independente de outras regras;
- *Modifiability*: em conseqüência da modularidade, velhas regras podem ser modificadas independentemente das outras;
- Transparência: habilidade do sistema para explicar as decisões e soluções.

Entre as desvantagens podemos mencionar:

1. A ineficiência na execução do programa;
2. Opacidade: o conhecimento algorítmico não é manifestado naturalmente;
3. O conhecimento é expresso num conjunto desordenado e não estruturado de regras;
4. Semanticamente não aproveita as estruturas que o domínio do problema possui na realidade.

Representação do Conhecimento em FER

A seleção de um formalismo para expressar o conhecimento de um dado domínio é um problema básico. Assim, a eleição de um determinado método de representação possui uma clara inter-relação com o tipo de conhecimento a ser representado.

Devido às características próprias das primitivas do modelo MER, o formalismo utilizado na representação de conhecimento na FER são as regras de produção. Esta sintaxe é adequada ao tipo de conhecimento, dado que precisamos expressar as restrições ou condições sobre as quais o processo de identificação das primitivas do MER (entidades, atributos e relações) pode ser aplicado. Regras de produção são uma coleção de asserções em forma implicacional, fáceis de entender e de alterar. As ações das regras podem ser simples, como por exemplo assegurar que alguma proposição seja verdadeira, ou complexas, como modificar as regras ou o ambiente através de operações de Entrada-Saída (E-S).

As regras são do tipo:

Se **situação corrente** Então **ações**

Estrutura da Base de Conhecimentos

As regras na base de conhecimentos são da seguinte forma:

identificador de regra Se **situação corrente** Então **ações**
ou
identificador de regra Se **Antecedente** Então **Conseqüente**

Na descrição da estrutura de regras, tem-se que o **identificador de regras** deve ser um átomo no sentido da linguagem de programação PROLOG [Clo 84], isto é, um elemento reconhecido como único pelo sistema, e que geralmente possui a forma "regra_XXX". **Antecedente** é o conjunto de fatos associados de forma disjuntiva (conectivo "e") que devem ser satisfeitos para que a regra seja considerada válida. **Conseqüente** é o conjunto de fatos, associados em forma disjuntiva, que são validados pela máquina de inferência (sempre que os antecedentes sejam válidos). Assim, estes fatos são executados (se eles implicarem em algum procedimento a ser executável), ou aplicados sobre a base de dados à medida que se progride numa inferência.

Na base de conhecimentos podemos distinguir três classes de regras, indicadas a seguir:

1. **Regras de Modificação**: estas regras estabelecem as mudanças executadas pelo assistente e que podem ser solucionadas sem intervenção do usuário. Exemplo:

```
Regra:
 Se existe tipo(entidade,Valor) definido
      e esse Valor participa de um relacionamento definido por
           tipo(relação,Valor,Tipo,Relacionamento).
      e existe tipo(relacionamento,Relacionamento) definido
      e não existe outra relação da qual Relacionamento faz parte
Então
      Cria-se uma entidade abstrata 'sem nome' com cardinalidade default 1
           tipo(entidade,'sem nome').
           tipo(relação,'sem nome',1,Relacionamento).
```

   Nesta regra, podemos observar como o assistente inspeciona a base de dados e determina se existe um relacionamento que interage somente com uma entidade. Se existir, o assistente cria uma entidade fictícia "sem nome". Esta entidade assim criada receberá seu nome quando, através de alguma regra de erro, o assistente indicar ao usuário que é necessária a definição da entidade fictícia.

2. **Regras de Erros**: proporcionam sugestões ao usuário para completar a representação no caso de incorreções e incompletude. Estas regras chamam procedimentos para obter a informação que está faltando. Juntamente com algumas regras de erros, são mostradas certas sugestões, através das regras de informação. Exemplo:

```
Regra:
Se
   existe tipo(entidade,'sem nome') definida
Então
   erro('defina uma entidade')
```

3. **Regras de Informação**: definem os princípios e práticas contrárias à metodologia. Ainda utilizando o exemplo anterior, quando a regra de erro é disparada pela máquina de inferência, também é acionada a regra de informação que diz:

```
Regra:
Se
   um objeto da realidade é uma entidade
Então
   este é um objeto da realidade sobre o qual se deseja possuir informação
```

   Assim, o usuário pode ter uma idéia do tipo de erro e da maneira de solucioná-lo, através da explicação dos princípios da metodologia.

4. **Fatos** *default*: utilizados para definição dos valores de cardinalidade entre entidades e relacionamentos. Este tipo de fato permite fornecer os valores (*default*) da cardinalidade quando essa informação não estiver disponível. O assistente verifica, junto com o usuário, a validade da cardinalidade atribuída.

Pela descrição das regras pode-se ver que algumas delas são "ativas", isto é, ao serem aplicadas modificam o estado dos objetos e, portanto, o desempenho (como as regras de modificação e erro). As regras de informação são fundamentalmente "passivas". Sua função é a de proporcionar sugestões; elas não modificam o estado da especificação, nem o desempenho do assistente.

5.2 Base de Dados

A base de dados é a parte do assistente que contém o conhecimento sobre a aplicação que está sendo especificada. Na realidade, é um armazenamento do que foi feito em relação às transformações realizadas, desde as especificações em linguagem natural, até a sua representação por meio de primitivas da metodologia MER.

Uma forma normalmente utilizada é o par < atributo-valor>, que tem a forma:

$$< \textit{atributo} > \textbf{predicado} < \textit{valor} >$$

Por exemplo, o fato

entidade **tem** *nome*

pode ser implementado através do seguinte predicado:

predicado(< *atributo* >, < *valor* >)

Dessa forma, a análise da sentença

A companhia tem cem empregados

produz a seguinte interpretação

E1= companhia
E2= empregado
Tipo1= 1
Tipo2= N
Rel= tem

onde as seguintes primitivas são geradas na base de dados:

tipo(entidade,companhia).
tipo(entidade,empregado).
tipo(relacionamento,tem).
tipo(relação,companhia,1,tem).
tipo(relação,empregado,'N',tem).
tipo(par_ordenado,companhia,tem,empregado,1,'N').

Durante a consulta da base de conhecimentos, as seguintes regras são ativadas

regra :

Se duas entidades possuem cardinalidade 1 e N com um relacionamento **então** o relacionamento é do tipo "um para muitos"

regra :

Se duas entidades possuem cardinalidade N e 1 com um relacionamento **então** o relacionamento é do tipo "muitos para um"

dando lugar à geração dos seguintes predicados:

tem é um_para_muitos.
tem é muitos_para_um.

Portanto, vemos que a base de dados contém dois tipos de predicados: um que identifica as primitivas do modelo e outro que identifica o tipo de relacionamento.

tipo(entidade, Nome_da_entidade).

tipo(relacionamento, Nome_do_relacionamento).

tipo(atributo, Nome_do_atributo).

tipo(possui, Nome_da_entidade, Nome_do_atributo).

tipo(relação, Nome_da_entidade, Cardinalidade, Nome_do_relacionamento).

tipo(par_ordenado, Entidade1, Relacionamento, Entidade2, Cardinalidade1, Cardinalidade2).

Relacionamento é um_para_um.

Relacionamento é um_para_muitos.

Relacionamento é muitos_para_um.

Relacionamento é muitos_para_muitos.

Relacionamento é ternário.

Relacionamento é ternário_um_para_um.

Relacionamento é ternário_muitos_para_muitos.

5.3 Máquina de Inferência

5.3.1 Manipulação do Conhecimento

Realizar uma inferência em um sistema baseado em regras de produção significa examinar a base de dados determinando que regras aplicar. Para que as inferências sejam realizadas necessita-se de um mecanismo de controle, conhecido como estratégia de controle, que deve possuir as seguintes características [Kvi 88]:

- Ser exaustivo;
- Ser sistemático: para um estado determinado do sistema, gera uma vez só a base de dados;
- Ser eficiente: evita o problema da explosão combinatória.

A máquina de inferência pode selecionar as regras a serem aplicadas à base de dados de três maneiras: encadeamento direto, encadeamento reverso e encadeamento misto.

Estratégia de Controle por Encadeamento Direto

Nesta estratégia de controle, a máquina de inferência examina os antecedentes de todas as regras da base de conhecimentos e seleciona a regra que tenha os seus antecedentes (parte < condição >) satisfeitos pelos fatos da base de dados. Se a execução não dar origem a chamada de outras regras ou fatos, termina-se a inferência; caso contrário, o processo de inferência continua até que seja obtida a solução do problema, ou até que não se tenha mais regras a serem aplicadas à base de dados.

Estratégia de Controle por Encadeamento Reverso

Nesta estratégia, quando uma ação é executada, a máquina de inferência procura obter uma base de dados que corresponda ao objetivo associado à solução do problema.

A máquina de inferência seleciona uma ou várias regras da base de conhecimentos, cujo consequente (< ação >) coincide com o objetivo procurado, formando assim um conjunto [Naz 91]. Deste conjunto, escolhe-se uma das regras e verifica-se sua validade com relação à base de dados. Se seus antecedentes forem válidos, a regra é executada; senão, a máquina de inferência seleciona uma outra regra do conjunto. Este processo é repetido até que não seja possível obter regras que satisfaçam o objetivo e/ou sub-objetivos (quando algumas sub-condições tenham que ser provadas).

Estratégia de Controle por Encadeamento Misto

Esta estratégia utiliza uma combinação das duas estratégias descritas anteriormente. A máquina de inferência combina as conclusões obtidas por encadeamento direto com os sub-objetivos obtidos por encadeamento reverso. A idéia principal deste tipo de encadeamento é resolver o problema da explosão combinatorial, diminuindo o espaço de busca.

5.3.2 Estratégias de Manipulação do Conhecimento na FER

A estratégia de controle utilizada pela ferramenta é o controle por encadeamento direto. A escolha dessa estratégia de inferência justifica-se devido à tarefa de análise ser inerentemente um processo de diagnóstico. O encadeamento direto analisa os fatos e obtém as conclusões destes fatos. Embora ambos os tipos de encadeamento (direto ou reverso) possam ser aplicados nesta tarefa, no encadeamento reverso deve-se "assumir algum diagnóstico provável", e comprovar se a base de dados o confirma. O encadeamento direto trabalha de forma mais natural, analisando fatos e tirando conclusões.

5.4 Interface em Linguagem Natural

5.4.1 Tipos de Gramáticas

Para implementar uma interface que receba respostas em linguagem natural, é necessário determinar uma gramática que permita organizar as palavras da linguagem em frases.

Formalmente, uma linguagem é um conjunto de sentenças, onde cada sentença é uma cadeia de um ou mais símbolos (palavras) do vocabulário da linguagem. Uma gramática é uma especificação finita e formal deste conjunto. Uma gramática pode tomar muitas formas. Nesta seção apresentaremos um resumo dos vários formalismos gramaticais lógicos propostos para interpretar a linguagem e extrair o seu significado.

Gramaticas Lógicas

Gramáticas lógicas foram desenvolvidas no contexto de sistemas de dedução por computador como um método de expressar gramáticas em lógica. A idéia é separar a especificação lógica do problema a ser resolvido do procedimento que interpreta essa especificação lógica, para resolver o problema.

Gramáticas de Cláusulas Definidas

As Gramáticas de Cláusulas Definidas (DCGs) [Per 80] surgiram para resolver o problema do contexto intrasentencial de gênero e número nas Gramáticas Livres de Contexto (CFGs). DCGs são uma extensão das CFGs e fornecem mecanismos para dois aspectos importantes da análise da linguagem:

1. Construção de estruturas (sintáticas ou semânticas): um aspecto no qual as DCGs são uma extensão das CFGs é na possibilidade que os não terminais possuam argumentos. É comum usar estes argumentos com o propósito de construir estruturas enquanto se está fazendo a análise de uma cadeia. A gramática da Figura 5.2, além de fazer a análise sintática, gera a árvore de derivação correspondente à sentença "Hamlet ama Gertrudes", que é mostrada na Figura 5.3.

```
sentença(s(SN,SV)) --> sintagma_nominal(SN),sintagma_verbal(SV).

sintagma_nominal(sn(P)) --> nome_proprio(P).

sintagma_verbal(sv(V,SN)) --> verbo(V),sintagma_nominal(SN).

nome_proprio(p(hamlet)) --> [hamlet].

nome_proprio(p(gertrudes)) --> [gertrudes].

verbo(v(ama)) --> [ama].
```

Figura 5.2: Uma Simples DCG

A primeira regra da Figura 5.2 pode ser lida como: A sentença com interpretação s(SN,SV) consiste de um sintagma[1] nominal com interpretação SN e de um sintagma verbal com interpretação SV.

2. Manipulação de dependências de contexto: Os argumentos não terminais podem ser usados para checar as limitações dependentes do contexto sobre as formas permitidas dos não terminais. O lado direito da regra pode conter, além dos símbolos terminais e não terminais, um outro tipo de ítem — os testes extras — que especificam as condições que devem ser satisfeitas pela regra DCG para ser validada no contexto. Estes testes extras são escritos entre chaves para indicar que são metas PROLOG e que não devem ser alterados pelo mecanismo que traduz regras DCG em cláusulas definidas.

DCGs podem ser suportadas de diversas maneiras dentro do ambiente PROLOG. Usualmente as regras DCG são compiladas em cláusulas definidas PROLOG. Quando uma DCG é interpretada,

[1]Chama-se sintagma um grupo de elementos lingüísticos que formam uma unidade numa organização hierarquizada. O termo sintagma é seguido de um qualificativo que define sua categoria gramatical (sintagma nominal, sintagma verbal, sintagma adjetival, etc.) [Dub 83]

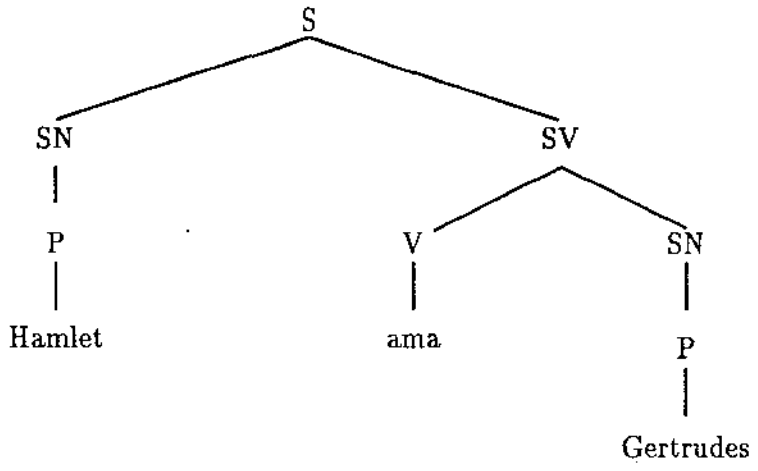


Figura 5.3: Árvore de Derivação para a Sentença "Hamlet ama Gertrudes"

cada regra na DCG é mapeada numa cláusula definida correspondente. Para cada não terminal de n argumentos na regra DCG, associa-se um predicado com $n+2$ argumentos. Os argumentos do predicado representam os pontos de início e de fim na cadeia de palavras, sendo analisada por este não terminal. Por exemplo, a regra

sentença(s(SN,SV)) −− > sintagma_nominal(SN), sintagma_verbal(SV).

é traduzida para cláusulas definidas do tipo:

sentença(s(SN,SV),S0,S) −− > sintagma_nominal(SN,S0,S1), sintagma_verbal(SV,S1,S).

Essa regra pode ser lida da seguinte forma: A lista de palavras S0 menos a lista de palavras S constitui uma sentença com interpretação s(SN,SV) se for possível encontrar uma lista intermediária de palavras S1 tal que a lista de palavras S0 menos a lista intermediária S1 é um sintagma nominal com interpretação SN, e a lista intermediária S1 menos a lista de palavras S é um sintagma verbal com interpretação SV.

Gramáticas de Extraposição

Gramáticas de Extraposição (XGs) são uma extensão das DCGs e similarmente definidas em termos de cláusulas lógicas [Per 80]. XGs foram desenvolvidas com o objetivo de tratar o fenômeno da extraposição à esquerda de uma maneira automática. Considere a seguinte sentença:

O empregado [que$_i$ o gerente despediu[t]$_i$] protestou.

A cláusula relativa foi colocada entre colchetes e a posição da qual o pronome relativo "que" foi movido é indicada pela posição do t indexado, que é chamado de *trace*.

Durante a análise da frase anterior, quando encontramos a palavra "que" entendemos que:

- As palavras imediatamente à direita devem ser analisadas como uma sub-sentença;

- A análise desta sub-sentença falhará devido à ausência de um sub-constituinte;
- Podemos prevenir a falha usando uma cópia da frase que está imediatamente à esquerda da palavra "que", ou seja "o empregado", para preencher a ausência do sub-constituinte.

Consideremos a gramática da Figura 5.4. Todas as regras, exceto a regra (8), são regras DCG.

(1) sentença --> sintagma_nominal,sintagma_verbal.
(2) sintagma_nominal --> artigo,nome,relativa.
(3) sintagma_nominal --> trace.
(4) sintagma_verbal --> verbo,sintagma_nominal.
(5) sintagma_verbal --> verbo.
(6) relativa --> marca,sentença.
(7) relativa --> [].
(8) marca...trace --> pronome_relativo.
(9) artigo --> [D],{e_artigo(D)}.
(10) artigo --> [].
(11) nome --> [N],{n(N)}.
(12) pronome_relativo --> [R],{e_pro(R)}.
(13) verbo --> [V],{v(V)}.
(14) e_artigo(a).
(15) n(mulher).
(16) e_pro(que).
(17) v(vive).
(18) v(le).

Figura 5.4: Uma Simples XG

Na regra

marca...trace--> pronome_relativo.

podemos expandir "marca" como um pronome_relativo na condição que alguma categoria à direita do pronome relativo, que não é realizada em toda cadeia terminal, possa ser expandida por um

trace. Não é difícil perceber que as regras (6) e (8) na Figura 5.4 vão aceitar sentenças agramaticais como:

A mulher [que_i o homem lê o livro] lê [t_i].

A gramática utilizará o *trace* introduzido pelo "que" para completar o objeto vazio do sintagma nominal na segunda ocorrência do verbo "ler". Isto claramente é um erro que levou à incorporação de uma restrição chamada de "restrição de colchetes". Colchetes são introduzidos para evitar que relações entre constituintes de fora e de dentro da cláusula relativa sejam efetuadas. Para introduzir esta restrição, foi necessário modificar a regra (6) para

relativa $--> $ abre,marca,sentença, fecha.

e introduzir a seguinte regra:

abre...fecha $--> $ [].

Os não terminais *abre* e *fecha* evitam o reposicionamento de qualquer constituinte da parte de fora da sub-sentença para dentro dela.

Gramáticas de Estruturas Modificadas

As Gramáticas de Estruturas Modificadas (MSGs) foram desenvolvidas por Dahl e McCord [Dah 83], com o propósito de resolver o problema da coordenação nas XGs. Coordenação são as construções gramaticais com conjunções (e,ou,mais) que têm sido um dos fenômenos de linguagem natural mais difíceis de serem manipulados porque podem envolver uma variedade muito grande de constituintes gramaticais (ou fragmentos não constituintes); além disso, elipses (ou reduções) podem ocorrer nos itens unidos por conjunções [Dah 83], ocasionando dificuldade no processamento.

A sintaxe das MSGs é similar à sintaxe das XGs. Regras XG são regras válidas em MSGs; entretanto, as MSGs também possibilitam a utilização de um termo chamado ítem semântico *Sem*. O ítem semântico possui um papel importante na interpretação semântica da frase analisada. Por exemplo na frase

João come uma maçã e uma pera.

a palavra "e" nos indica que neste ponto um *backup* deve acontecer com o objetivo de analisar as palavras que seguem a conjunção em paralelo a alguma categoria frasal encontrada previamente; neste caso " uma pera" é analisada em paralelo a " uma maçã", gerando-se duas sub-árvores de derivação que serão unidas numa árvore de derivação final. O ítem semântico *Sem* aparece do lado esquerdo da regra gramatical e é da forma:

$$a : Sem --> b.$$

Quando a regra MSG não traz consigo um *Sem* explícito, o sistema insere um *Sem* especial *l-true*. Por exemplo, na regra

$$a --> b$$

o sistema insere o ítem *l-true*; portanto, a regra inicial é transformada em:

$$a:\ l\text{-}true --> b.$$

A sentença

Cada homem comeu uma maçã e uma pera.

é corretamente analisada pela gramática da Figura 5.5 sendo produzida a seguinte formula lógica:

cada(X,homem(X),existe(Y,maça(Y),comeu(X,Y)& existe(Y,pera(Y),comeu(X,Y)))

Entretanto, sentenças agramaticais como

Cada homem comeu uma e uma pera

são também aceitas.

Podemos concluir, entretanto, que a idéia de usar análise paralela envolvendo coordenação é muito útil; porém, um controle deve ser exercido sobre a ordem de execução das cláusulas para permitir que somente certas categorias frasais sejam usadas pelo analisador paralelo.

Gramáticas de Extraposição e Anáfora (AXGs)

As MSGs não resolvem o problema de referência pronominal. Anáforas são referências abreviadas a expressões que apareceram antes no contexto ou na frase. Seja por exemplo:

Quem defendeu Maria da ira do padrasto que a odiava?

O pronome "a" que aparece na expressão "que a odiava" é um exemplo de anáfora, onde seu referente é Maria. Uma solução proposta por Carvalho [Car 89] é o formalismo gramatical Gramáticas de Extraposição e Anáfora (AXGs), que podem ser consideradas extensões das XGs, com tratamento de referência pronominal.

Durante a análise da frase anterior, quando encontramos a letra "a", existe uma busca por um referente feminino; como Maria preenche os requisitos de gênero e número, o pronome "a" é dito se referir a Maria.

Anáforas podem ser manipuladas pela regra AXG

$$a//b - - > c.$$

que pode ser lida como: Um a que introduz uma marca referencial b pode ser reescrito como um c. Regras XG do tipo

$$a...b - -- > c, d.$$
$$a - - > b, c, d.$$

são regras válidas em AXGs.

Além das regras gramaticais, AXG possui declarações de domínio de co-referência, que tomam a forma de:

@α, com α tomando valores na classe de símbolos não terminais V_N.

Estas declarações meta-gramaticais são usadas quando restrições envolvendo a noção de domínio co-referencial são encontradas.

```
sentença --> sintagma_nominal(X),sintagma_verbal(X).

sintagma_nominal(X) --> artigo(X),nome(X).

sintagma_verbal(X) --> verbo(X,Y),sintagma_nominal(Y).

artigo(X):Sem --> [D],{e_artigo(D,X,Sem)}.

nome(X):l-Pred --> [N],{n(N,X,Pred)}.

verbo(X,Y):l-Pred --> [V],{v(V,X,Y,Pred)}.

e_artigo(cada,X,P/Q-cada(X,Q,P)).

e_artigo(um,X,U/V-existe(X,V,U)).

n(maçã,X,maçã(X)).

n(pera,X,pera(X)).

n(mulher,X,mulher(X)).

n(homem,X,homem(X)).

v(comeu,X,Y,come(X,Y)).

conj(e,conj(e),S*T-(S&T)).
```

Figura 5.5: Uma Simples MSG

5.4.2 Gramática Eleita

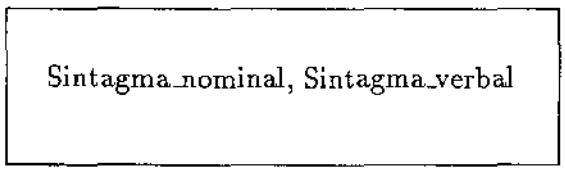
Uma vez estudados os formalismos gramaticais lógicos DCGs, XGs, MSGs e AXGs e estabelecidas as vantagens com relação às CFGs, vamos definir nesta seção o formalismo gramatical utilizado pela interface do assistente especialista. Vimos que tanto XGs, como MSGs e AXGs são extensões do formalismo DCG, que tendem a abranger o tratamento específico de algum problema num sentido mais rigoroso. Portanto, decidimos optar pelo formalismo DCG. A principal diferença entre DCGs e suas extensões (XGs,MSGs,AXGs) está no fato de que tanto XGs como MSGs e AXGs são gramáticas de implementação automática [Car 89], isto é, quem escreve a gramática não tem que se preocupar com restrições, pois elas são automaticamente implementadas pelos formalismos. No entanto, numa gramática DCG, é necessário estabelecer as regras necessárias para tratar todos os casos previstos pela gramática.

Para poder definir o tipo de regras que compõem a gramática, necessitamos levar em consideração os seguintes aspectos:

1. Tipos de orações a serem analisadas pela gramática;

2. Como será efetuada a análise sintática.

1. Tipos de orações: para decidir os tipos de sentenças que a gramática deverá aceitar, nós nos baseamos na semântica do MER, onde a mínima representação de uma aplicação deve ser composta de um relacionamento binário. Portanto, necessitamos que o usuário transmita a informação através de sentenças gramaticais da forma



onde o verbo pode ser intransitivo, transitivo direto e/ou indireto, verbo composto e verbo de ligação. No caso do verbo ser intransitivo é necessário um complemento (por exemplo adjunto adverbial), para podermos definir a segunda entidade. Um verbo bitransitivo é uma forte indicação da presença de duas entidades (sujeito e objeto) e um relacionamento (verbo transitivo); um verbo de ligação é uma forte indicação da presença de um atributo (objeto) de uma entidade ou relacionamento (sujeito). Estas restrições se baseiam na correspondência entre as estruturas sintáticas e a semântica dos objetos no MER (vide capítulo 4).

Exemplo:

Pacientes inteligentes marcam consultas.

A conta é individual.

Outros tipos de sentenças gramaticais aceitas são:

1. O secretário e o chefe de seção contratam os funcionários.
2. O secretário analisa o curriculum e contrata os funcionários.
3. A secretária atende o paciente, cobra a consulta e cuida do consultório.
4. O secretário que trabalha com o chefe de seção analisa o curriculum.
5. O secretário, o chefe de seção e o chefe de pessoal contratam os funcionários.
6. O atendimento de consultas é rápido, agradável e correto.
7. A companhia tem cinqüenta sedes.
8. Cada departamento tem um relógio de ponto que registra os horários.
9. As cinquenta sedes localizam-se em quarenta estados.
10. Um tipo de trabalho pode ser executado em várias estações de trabalho.
11. O consultório precisa de uma lanchonete.
12. A secretária com os médicos comeriam na lanchonete.

Devido ao tipo das sentenças gramaticais aceitas, vemos a necessidade de definir o tratamento de:

- **Pronome Relativo:** encontrado nas sentenças 4 e 8, através das seguintes cláusulas

```
sentença(X,Entidade1,Tipo) -- >
    sintagma_nominal(Gen,Num,X,Tipo,Entidade1),
    sintagma_verbal(Gen,Num,sx,Relacionamento,Tipo2,Entidade1,Entidade2).
```

onde sx significa sem direito a usar sintagma nominal extraposto [Sav 88]; dessa maneira controlamos o nome extraposto para que ele seja usado uma única vez, ou para substituir o agente, ou ainda para substituir algum outro componente.

- **Concordância de Gênero e Número**

- **Coordenação**: o tratamento da coordenação é bem mais simples que o utilizado pela MSG e é efetuado através das seguintes cláusulas

```
sentença(X,Entidade1,Tipo) -- > sintag_nominal(Gen,Num,X,Tipo,Entidade1),
   sintag_v_comp(Gen,Num,sx,Relacionamento,Tipo2,Entidade1,Entidade2)
sentença(X,Entidade1,Tipo) -- >
   s_composto(Gen,Num,sx,Tipo,Entidade1),
   sintagma_verbal(Gen,Num,X,Relacionamento,Tipo2,Entidade1,Entidade2).
```

onde sintag_v_comp(Gen,Num,sx,Relacionamento,Tipo2,Entidade1,Entidade2) é uma regra para tratamento de sintagma verbal composto formado por vários sintagmas verbais onde todos são separados por vírgulas, menos o último sintagma verbal, que é precedido de um "e". S_composto(Gen,Num,sx,Tipo,Entidade1) é uma regra para tratamento de sintagma nominal composto formado por vários sintagmas nominais onde todos são separados por vírgulas, menos o último sintagma nominal, que é precedido de um "e".

2. Como será efetuada a análise sintática: A análise sintática pode ser implementada de duas formas: descendente (ou de cima para baixo), começando pelo símbolo inicial e aplicando as regras gramaticais até que os símbolos não terminais da árvore correspondam aos componentes da sentença que estiver sendo analisada, ou ascendente (de baixo para cima), onde todas as regras começam com o reconhecimento de palavras (permitindo abandonar a análise se a palavra não existir no dicionário).

Optamos por utilizar a forma descendente já que o problema da falta de palavras no dicionário é resolvido fazendo-se uma busca no léxico antes de realizar qualquer análise gramatical da frase, permitindo ao usuário definir alguma palavra que o assistente não conheça e indicando sua categoria gramatical.

5.5 Conclusão

Neste capítulo, foi apresentada a arquitetura do assistente especialista e discutiram-se as técnicas adotadas na:

- Construção da base de conhecimentos;
- Estrutura da base de dados;
- Estratégia utilizada pela máquina de inferência;
- Gramática utilizada na especificação de requisitos.

Capítulo 6

Exemplo da Aplicação

O objetivo deste capítulo é mostrar as principais características da ferramenta desenvolvida. Na primeira seção, são apresentadas as opções da ferramenta, descrevendo-se os seus principais elementos. Na segunda seção, é apresentado um exemplo de interação efetuada em linguagem natural, entre o usuário e o assistente, com o respectivo modelo textual gerado.

6.1 Módulos da Ferramenta

A ferramenta foi desenvolvida utilizando-se o PROLOG de *Edinburgh*, versão 1.5.04 [Bow 87] que, por ser uma versão simples, necessitou a criação de várias rotinas; isso facilita que a ferramenta seja utilizada em outros ambientes, como por exemplo no ARITY PROLOG [Ari 87].

Quando a interação entre o usuário e a ferramenta tem início, um menu com as seguintes opções é apresentado:

- Consultar;
- Editar;
- Ler Parágrafo,
- Ler Primitivas;
- Verificar;
- Gerar;
- Começar;
- Terminar.

Na Figura 6.1 podemos ver cada um desses itens, com as funções realizadas pela FER.
A opção Consultar "chama" o assistente através dos processos de:

- Especificação: onde são apresentadas as perguntas, obtidas as respostas e gerada a base de dados;

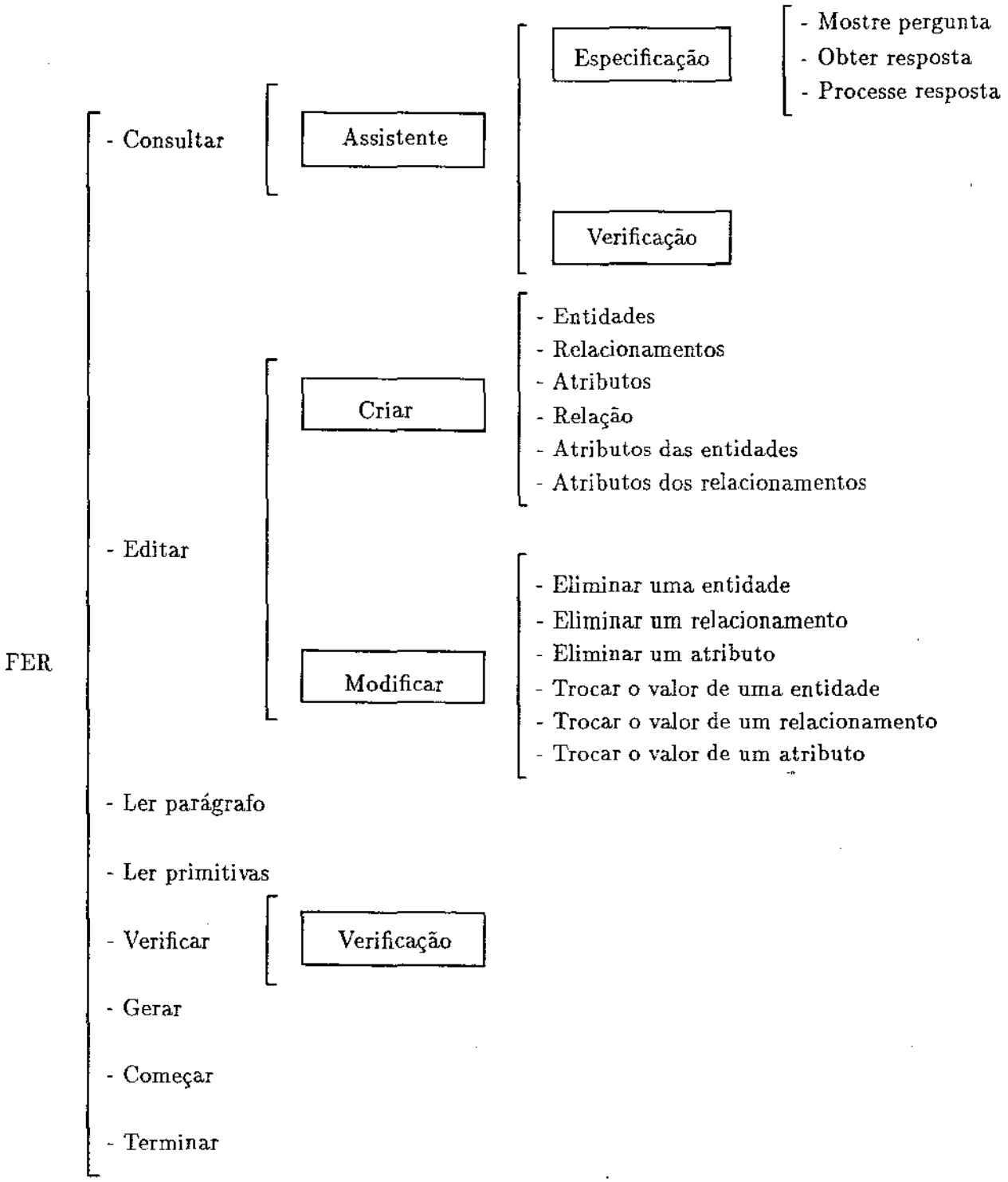


Figura 6.1: As Opções do Menu da FER

- Verificação: onde a base de conhecimentos é consultada. À medida que a base está sendo consultada, são indicadas as regras que foram disparadas, juntamente com uma explicação das mesmas. Isso é feito para que o usuário se informe sobre a metodologia.

Em caso do usuário responder erroneamente a uma pergunta, uma explicação da mesma é apresentada. Se o assistente não conhecer uma palavra, ele pedirá ao usuário que a incorpore ao sistema indicando a sua categoria gramatical.

Quando o usuário seleciona a opção Editar, FER mostra as possibilidades de criar ou modificar as primitivas do MER de forma automática. Contudo, para utilizar essa opção é preciso conhecer a semântica do MER.

A opção Ler Parágrafo permite obter uma especificação de uma ou mais sentenças, separadas por ponto e analisá-las gramaticalmente. Esta opção é utilizada para aqueles casos onde não se deseja responder as perguntas, mas fazer a especificação através de orações.

A opção Ler Primitivas permite obter uma especificação criada anteriormente e armazenada num arquivo no formato das primitivas da "meta-linguagem". Esta opção é utilizada para quando já se tem uma especificação.

A opção Verificar chama o assistente através da opção Verificação. Esta opção é usada quando se deseja verificar as primitivas obtidas de forma automática, isto é, utilizando-se a opção Criar, estão corretas.

A opção Gerar mostra o modelo textual ao usuário.

A opção Começar elimina o conteúdo da base de dados.

Finalmente, a opção Terminar permite a saída do sistema.

6.2 Exemplo

O exemplo mostrado a seguir é uma descrição de requisitos de informação de uma firma industrial. Este estudo de caso foi originalmente proposto por Chen no seu artigo *English Sentence Structure and Entity-Relationship Diagrams* [Che 83a]. Nas próximas seções são descritos o diálogo com o assistente, a transformação das respostas em português para primitivas do MER, a verificação do diagrama obtido (através da consulta à base de conhecimentos) e, por último, a apresentação do modelo textual correspondente.

6.2.1 Especificação

A especificação começa com um diálogo onde não é necessário responder a todas as perguntas apresentadas. A seguir são mostradas as perguntas feitas ao usuário, juntamente com as respostas obtidas.

1. A organização.

   (a) Indicar de que é composta a organização.
   Respostas:

   A companhia tem cinqüenta sedes.

   A companhia tem cem empregados.

   (b) Indicar o objetivo da organização.

(c) Indicar os fornecedores da organização e o que eles fornecem.

(d) Citar outros orgãos externos à organização com os quais ela possui algum relacionamento.

(e) Indicar que relatórios ou informações precisam ser passados para algum orgão externo.

2. Descrição funcional.

(a) Indicar a atividade básica da organização.

(b) Com relação a atividade principal da organização, indicar os dados que são utilizados ou produzidos durante essa atividade.

(c) Descrever informalmente como cada atividade (ou função) é realizada, e que parte da organização a realiza.

3. Características estruturais da organização.

(a) Indicar as partes que compõem a organização.
Respostas:

As cinqüenta sedes localizam-se em quarenta estados
Cada sede divide-se em departamentos
Os departamentos dividem-se em estações de trabalho

(b) Indicar os objetivos de cada parte.
Respostas:

Os empregados filiam-se a união dos trabalhadores
As cinco uniões de trabalhadores estão representadas na companhia.

(c) Se alguma parte da organização se relacionar com outra, indicar com quem e através de que atividade.
Respostas:

Um tipo de trabalho pode ser executado em várias estações de trabalho
Empregados podem trabalhar nas estações de trabalho

4. Clientes/Usuários da organização.

(a) Indicar quais os benefícios que os usuários recebem da organização.

(b) Indicar com que funções interagem os usuários.

(c) Dar sugestões em termos de atividade que os usuários gostariam de ter.

5. Fluxo de informação.

(a) Para cada atividade fundamental, descrever que dados utiliza, possui ou emite.
Respostas:

Cada departamento tem um relógio de ponto que registra os horários
A estação de trabalho possui um coletor de dados
Os empregados que utilizam o coletor de dados reportam as atividades

(b) Indicar documentos, informes ou outras saídas, e a que setor (ou parte da organização) pertencem.

(c) Indicar documentos e informes que pertencem a algum orgão externo.

(d) Propor uma atividade e indicar que informações ela requer para ser realizada.

(e) Listar os novos elementos de dados que os usuários gostariam de ver no novo sistema. Indicar também de onde surgem e onde são utilizados.

6. O futuro da organização.

(a) Indicar, segundo o seu critério, que novas seções deveriam existir dentro da organização, especificando com quem deveriam se relacionar.

(b) Indicar o que fazem as novas seções apresentadas acima.

(c) Indicar que informações ou dados podem ser acresentados a alguma atividade já descrita ou a uma nova atividade, para melhorar o seu funcionamento.

(d) Indicar que novo serviço poderia ser prestado pela organização, dizendo que função executaria.

Interpretação das Sentenças

Para cada resposta obtida, o assistente verifica se as palavras da sentença são conhecidas. Se existir alguma palavra que o assistente desconhece, ele pede ao usuário para fornecer a categoria gramatical da palavra, que é incorporada ao dicionário. Além disso, o assistente consulta o usuário sobre o traço semântico do verbo de cada sentença, pois os traços semânticos dos nomes e do verbo devem ser compatíveis. Exemplo: "Maria morava em Campinas", onde o agente do verbo morar deve ser animado e o complemento do verbo deve ser um lugar. O assistente pergunta ao usuário se este utiliza formas compostas de verbos. Caso a resposta seja afirmativa, o assistente pede que seja fornecido o verbo principal e o verbo auxiliar (até dois verbos auxiliares) [Ari 92e].

Como toda resposta, uma vez processada gramaticalmente, pode ser dividida em no mínimo duas entidades, um relacionamento e dois tipos, então o resultado do processamento em linguagem natural é obtido em cinco variáveis, a saber: E1, E2, Rel, Tipo1 e Tipo2, onde o conteúdo dessas variáveis é baseado em regras como:

Um nome comum corresponde a um tipo entidade
Um verbo transitivo corresponde a um tipo relacionamento

Portanto, E1 contém o sujeito da resposta, Rel contém o verbo, E2 contém o objeto. Tais regras correspondem às heurísticas estabelecidas por Chen, que juntamente com algumas heurísticas de Loh, foram implementadas. As variáveis Tipo1 e Tipo2 contêm a cardinalidade da relação. Estas são definidas através de regras heurísticas por nós definidas, já que na literatura não existe referência a determinação automática das cardinalidades. A cardinalidade é estabelecida como resultado da presença de artigos ou referência a quantidades dentro da resposta. Como na maioria dos trabalhos, existe também uma consulta ao usuário para validar a cardinalidade.

Estas variáveis, que contêm a semântica do MER, serão posteriormente interpretadas a fim de gerar as primitivas da meta-linguagem. As variáveis E1, Rel e E2 podem conter operadores do tipo "&", "=>" e "*he*". O primeiro operador indica conjunção de nomes de entidades ou

relacionamentos, o segundo operador indica que o nome do lado direito é um atributo da entidade ou relacionamento do lado esquerdo. O operador "*he*" estabelece ligações do tipo "é parte de", onde nomes comuns são usados em combinação com o verbo ser e uma preposição. Exemplo:

"A seção de pessoal é parte da seção de consultas"

onde *parte* está estabelecendo uma relação entre a seção de pessoal e a seção de consultas.

Os operadores "&" e "*he*" foram criados para o tratamento de sentenças compostas e do tipo "é parte de"; o tratamento de preposições, pronome relativo e cardinalidade são também extensões do trabalho de Chen.

Veremos a seguir o resultado do processamento de cada resposta, a geração das primitivas e o diagrama entidade-relacionamento correspondente.

A análise das sentenças

1. A companhia tem cinqüenta sedes
2. A companhia tem cem empregados

produz a seguinte interpretação:

E1= companhia
E2= sede
Tipo1= 1
Tipo2= N
Rel= tem

E1= companhia
E2= empregado
Tipo1= 1
Tipo2= N
Rel= tem

onde as seguintes primitivas são geradas:

```
tipo(entidade,empregado).
tipo(entidade,companhia).
tipo(entidade,sede).
tipo(relacionamento,tem).
tipo(relação,sede,'N',tem).
tipo(relação,companhia,1,tem).
tipo(relação,empregado,'N',tem).
tipo(par_ordenado,companhia,tem,empregado,1,'N').
tipo(par_ordenado,companhia,tem,sede,1,'N').
```

O diagrama mostrado na Figura 6.2[1] corresponde a interpretação das sentenças 1 e 2.

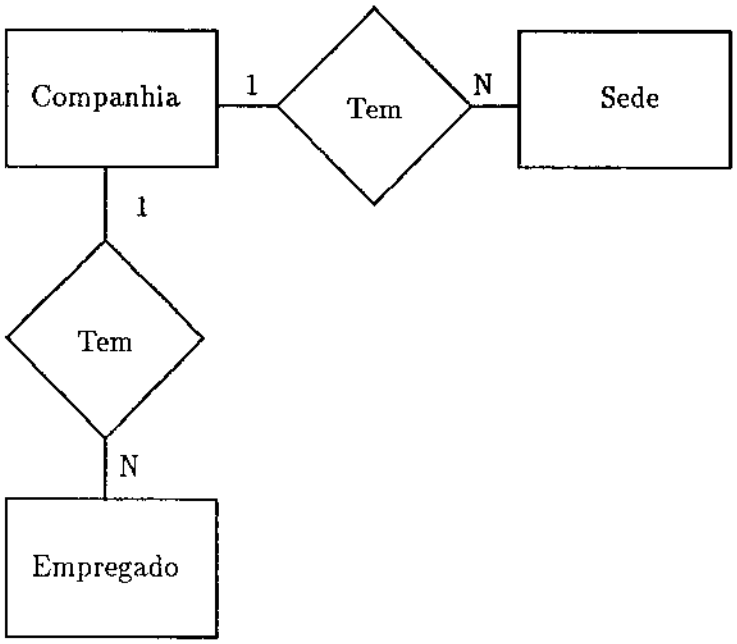


Figura 6.2: Diagrama ER das Sentenças 1 e 2.

A partir da análise da sentença

3. Cada departamento tem um relógio de ponto que registra os horários

a seguinte interpretação é gerada

E1= departamento

E2= relógio de ponto

Tipo1= 1

Tipo2= 1

Rel= tem

E1= relógio de ponto

E2= horário

Tipo1= 1

Tipo2= N

Rel= registra

[1]Em algumas figuras, existe discordância de número, pois o nome das entidades é sempre guardado no singular, evitando assim duplicações

com as seguintes primitivas:

tipo(entidade,relógio de ponto).
tipo(entidade,horário).
tipo(entidade,departamento).
tipo(relacionamento,registra).
tipo(relacionamento,tem).
tipo(relação,relógio de ponto,1,tem).
tipo(relação,departamento,1,tem).
tipo(relação,horário,'N',registra).
tipo(relação,relógio de ponto,1,registra).
tipo(par_ordenado,departamento,tem,relógio de ponto,1,1).
tipo(par_ordenado,relógio de ponto,registra,horário,1,'N').

O diagrama correspondente a essa análise é mostrado na Figura 6.3.

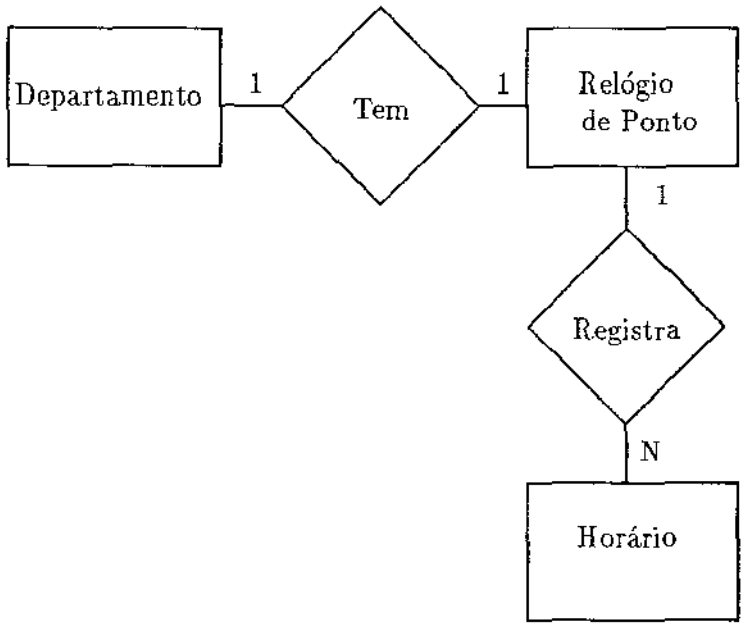


Figura 6.3: Diagrama ER da Sentença 3.

A partir das sentenças:

4. Os empregados filiam-se a união dos trabalhadores

5. As cinco uniões de trabalhadores estão representadas na companhia

a seguinte interpretação é gerada:

E1= empregado

E2= união de trabalhador

Tipo1= N

Tipo2= 1

Rel= filiam-se

E1= união de trabalhador

E2= companhia

Tipo1= N

Tipo2= 1

Rel= representadas

juntamente com as seguintes primitivas:

tipo(entidade,empregado).

tipo(entidade,união de trabalhador).

tipo(entidade,companhia).

tipo(relacionamento,representadas).

tipo(relacionamento,filiam-se).

tipo(relação,companhia,1,representadas).

tipo(relação,união de trabalhador,'N',representadas).

tipo(relação,união de trabalhador,1,filiam-se).

tipo(relação,empregado,'N',filiam-se).

tipo(par_ordenado,união de trabalhador,representadas,companhia,'N',1).

tipo(par_ordenado,empregado,filiam-se,união de trabalhador,'N',1).

O diagrama correspondente é mostrado na Figura 6.4.

A sentença

6. Um tipo de trabalho pode ser executado em várias estações de trabalho

produz a seguinte interpretação:

E1= tipo de trabalho

E2= estação de trabalho

Tipo1= 1

Tipo2= N

Rel= executado

e gera as seguintes primitivas:

tipo(entidade,estação de trabalho).
tipo(entidade,tipo de trabalho).
tipo(relacionamento,executado).
tipo(relação,estação de trabalho,'N',executado).
tipo(relação,tipo de trabalho,1,executado).
tipo(par_ordenado,tipo de trabalho,executado,estação de trabalho,1,'N').

O diagrama correspondente é mostrado na Figura 6.5.

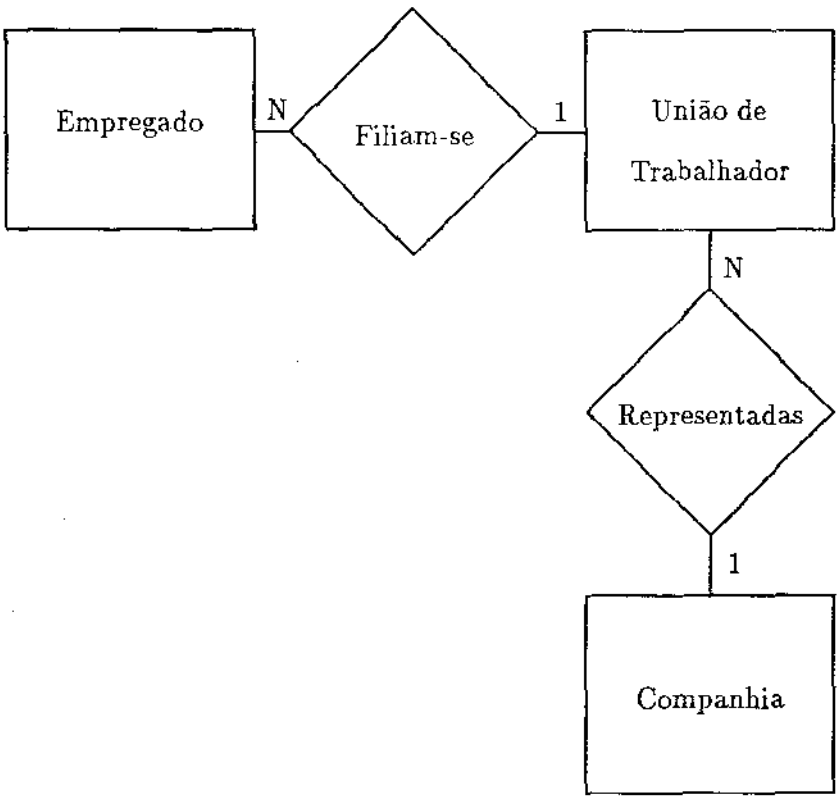


Figura 6.4: Diagrama ER das Sentenças 4 e 5.

As sentenças 7, 8 e 9:

7. As cinqüenta sedes localizam-se em quarenta estados
8. Cada sede divide-se em departamentos
9. Os departamentos dividem-se em estações de trabalho

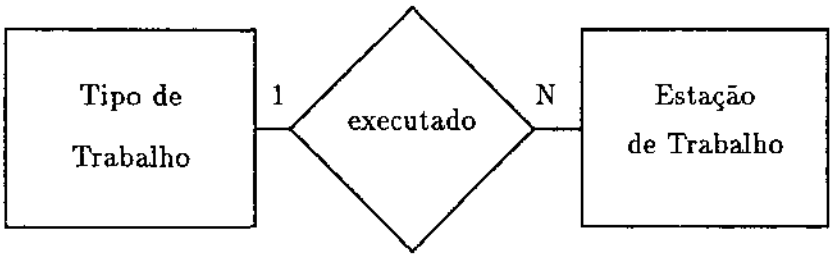


Figura 6.5: Diagrama ER da Sentença 6.

produzem a seguinte interpretação:

E1= sede
E2= estado
Tipo1= N
Tipo2= N
Rel= localizam-se

E1= sede
E2= departamento
Tipo1= 1
Tipo2= 1
Rel= divide-se

E1= departamento
E2= estação de trabalho
Tipo1= N
Tipo2= 1
Rel= dividem-se

e geram as seguintes primitivas:

tipo(entidade,sede).
tipo(entidade,estado).
tipo(entidade,departamento).
tipo(entidade,estação de trabalho).
tipo(relacionamento,dividem-se).
tipo(relacionamento,divide-se).

tipo(relacionamento,localizam-se).

tipo(relação,estação de trabalho,1,dividem-se).

tipo(relação,departamento,'N',dividem-se).

tipo(relação,departamento,1,divide-se).

tipo(relação,sede,1,divide-se).

tipo(relação,estado,'N',localizam-se).

tipo(relação,sede,'N',localizam-se).

tipo(par_ordenado,departamento,dividem-se,estação de trabalho,'N',1).

tipo(par_ordenado,sede,divide-se,departamento,1,1).

tipo(par_ordenado,sede,localizam-se,estado,'N','N').

O diagrama correspondente à interpretação das sentenças 7, 8 e 9 é mostrado na Figura 6.6.

As sentenças 10 e 11:

10. Empregados podem trabalhar nas estações de trabalho

11. Os empregados que utilizam o coletor de dados reportam as atividades

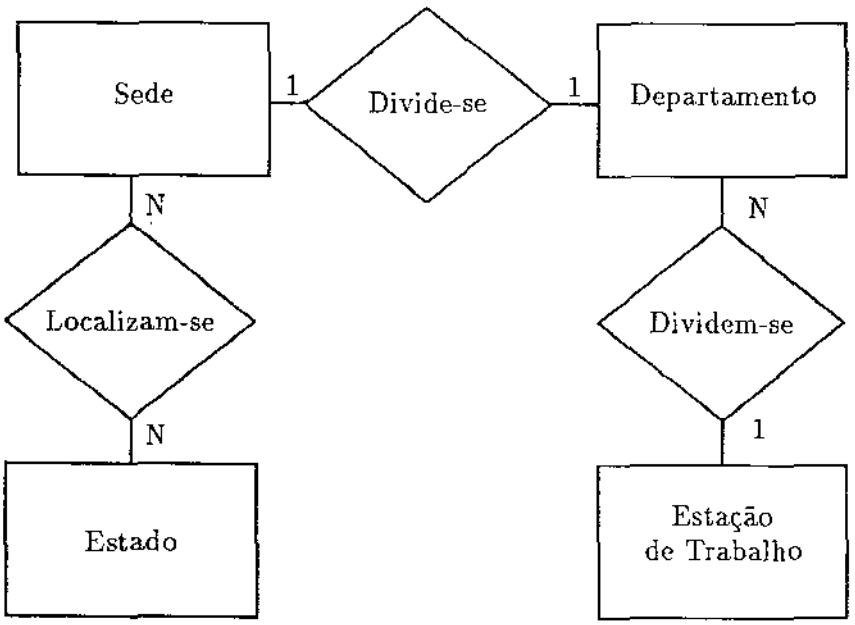


Figura 6.6: Diagrama ER das Sentenças 7, 8 e 9.

produzem a seguinte interpretação:

E1= empregado
E2= estação de trabalho
Tipo1= 1 Tipo2= N
Rel= trabalhar

E1= empregado
E2= coletor de dado
Tipo1= N Tipo2= 1
Rel= utilizam

E1= empregado
E2= atividade
Tipo1= N Tipo2= N
Rel= reportam

e geram as seguintes primitivas:

tipo(entidade,empregado).
tipo(entidade,estação de trabalho).
tipo(entidade,atividade).
tipo(entidade,coletor de dado).
tipo(relacionamento,reportam).
tipo(relacionamento,utilizam).
tipo(relacionamento,trabalhar).
tipo(relação,atividade,'N',reportam).
tipo(relação,empregado,'N',reportam).
tipo(relação,coletor de dado,1,utilizam).
tipo(relação,empregado,'N',utilizam).
tipo(relação,estação de trabalho,'N',trabalhar).
tipo(relação,empregado,1,trabalhar).
tipo(par_ordenado,empregado,reportam,atividade,'N','N').
tipo(par_ordenado,empregado,utilizam,coletor de dado,'N',1).
tipo(par_ordenado,empregado,trabalhar,estação de trabalho,1,'N').

O diagrama mostrado na Figura 6.7 corresponde à interpretação das sentenças 10 e 11.

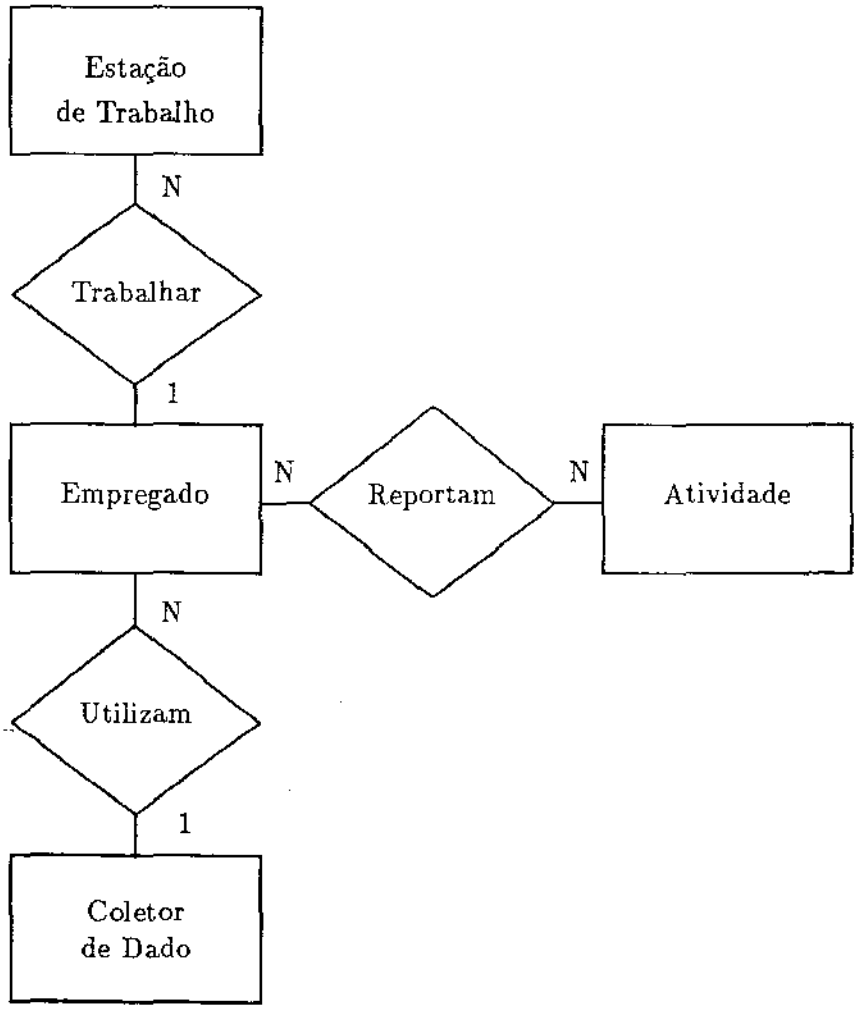


Figura 6.7: Diagrama ER das Sentenças 10 e 11.

A sentença

12. A estação de trabalho possui um coletor de dados

produz a seguinte interpretação:

E1= estação de trabalho

E2= coletor de dado

Tipo1= 1

Tipo2= 1

Rel= possui

e gera as seguintes primitivas:

tipo(entidade,estação de trabalho).

tipo(entidade,coletor de dado).

tipo(relacionamento,possui).

tipo(relação,coletor de dado,1,possui).

tipo(relação,estação de trabalho,1,possui).

tipo(par_ordenado,estação de trabalho,possui,coletor de dado,1,1).

O diagrama correspondente à análise dessas sentenças é mostrado na Figura 6.8.

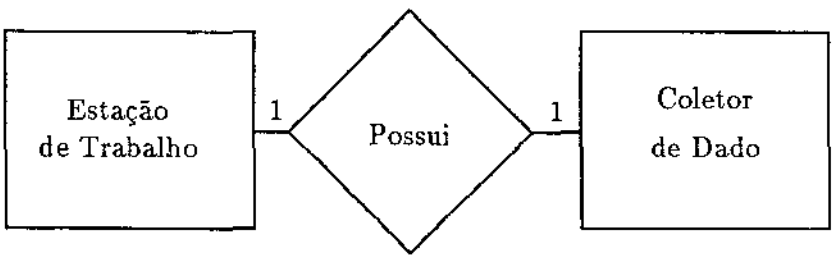


Figura 6.8: Diagrama ER da Sentença 12.

Finalmente, na Figura 6.9, é mostrado o diagrama completo da especificação.

6.2.2 Verificação

Uma vez obtida a interpretação das sentenças, começa a fase de verificação da especificação, onde a base de conhecimentos é utilizada. O objetivo deste processo é determinar inconsistências e erros, além de permitir ao usuário conhecer mais detalhes sobre a semântica do MER. A etapa de verificação valida, com o auxílio do usuário, a cardinalidade, as entidades, os relacionamentos e os atributos.

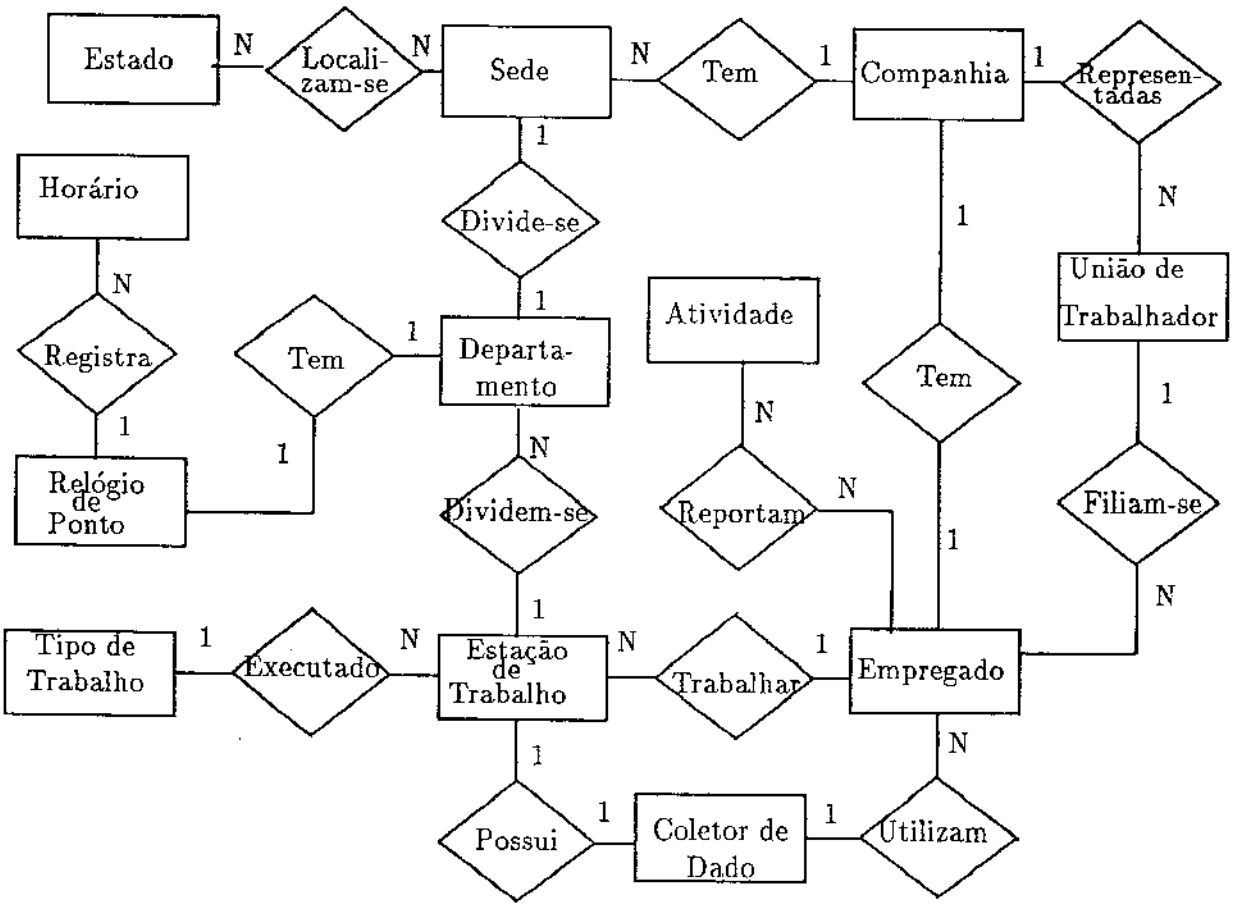


Figura 6.9: Diagrama ER Completo

A cardinalidade das relações foi mudada mediante colaboração do usuário, através do seguinte diálogo com o assistente:

Assistente: Processo de modificação da cardinalidade da seguinte relação: "sede se relaciona com estado através de localizam-se com tipos 1 e 1"
Assistente: Gostaria de redefinir as cardinalidades? s/n
Usuário: s.
Assistente: Indique a cardinalidade de sede
Usuário: 1.
Assistente: e de estado
Usuário: N.

A partir das respostas do usuário, o seguinte fato foi gerado:

tipo(par_ordenado,sede,divide-se,departamento,1,'N').

Da mesma forma, a cardinalidade das seguintes relações foi mudada

- tipo(par_ordenado,departamento,dividem-se,estação de trabalho,1,'N').
- tipo(par_ordenado,sede,localizam-se,estado,'N',1).

sendo que as seguintes regras foram utilizadas:

1. regra :

   Se duas entidades possuem cardinalidade N e M com um relacionamento
   então o relacionamento é do tipo "muitos para muitos"

2. regra :

   Se duas entidades possuem cardinalidade 1 e N com um relacionamento
   então o relacionamento é do tipo "um para muitos"

3. regra :

   Se duas entidades possuem cardinalidade 1 com um relacionamento
   então o relacionamento é do tipo "um para um"

4. regra :

   Se duas entidades possuem cardinalidade N e 1 com um relacionamento
   então o relacionamento é do tipo "muitos para um"

5. regra :

   Se existe uma entidade que não possui atributos
   então erro: defina seus atributos

6. regra :

   Se existe um relacionamento que não possui atributos
   então erro: defina seus atributos

7. regra :

   Se um conceito é um atributo
   então ele é uma característica ou propriedade do objeto

8. regra :

   Se um relacionamento é do tipo " um para muitos "
   então uma instância de uma entidade interage com várias instâncias da outra entidade

9. regra :

   Se um relacionamento é do tipo " um para um "
   então uma instância de uma entidade interage com uma instância da outra

10. regra :

    Se um relacionamento é do tipo " muitos para um "
    então várias instâncias de uma entidade interagem com uma instância da outra entidade

11. regra :

    Se um relacionamento é do tipo " muitos para muitos "
    então várias instâncias de uma entidade interagem com várias instâncias da outra

12. regra :

    Se um relacionamento interage com quatro entidades
    então erro: o relacionamento pode, no máximo, ser ternário

13. regra :

    Se um relacionamento interage com quatro entidades
    Então subdividi-se o relacionamento em dois relacionamentos

Através dessas regras, são geradas automaticamente as seguintes informações:

- "executado" é um_para_muitos;[2]
- "executado" é muitos_para_um;
- "filiam-se" é um_para_muitos;
- "filiam-se" é muitos_para_um;
- "representadas" é um_para_muitos;
- "representadas" é muitos_para_um;
- "reportam" é muitos_para_muitos;
- "utilizam" é um_para_muitos;
- "utilizam" é muitos_para_um;
- "trabalhar" é um_para_muitos;
- "trabalhar" é muitos_para_um;
- "tem-1" é um_para_um;
- "registra" é um_para_muitos;
- "registra" é muitos_para_um;
- "tem-1-1" é um_para_muitos;
- "tem-1-1" é muitos_para_um;
- "dividem-se" é um_para_muitos;
- "dividem-se" é muitos_para_um;
- "divide-se" é muitos_para_um;
- "divide-se" é um_para_muitos;
- "localizam-se" é um_para_muitos;
- "localizam-se" é muitos_para_um;
- "possui" é um_para_um;
- "tem" é um_para_muitos;
- "tem" é muitos_para_um;

[2]Um relacionamento que é "um para muitos" em um sentido é "muitos para um" no outro.

Os atributos das entidades e os relacionamentos que faltam definir na representação (regras cinco e seis), são obtidos mediante colaboração do usuário, através do seguinte diálogo com o assistente:

Assistente: Gostaria de definir os atributos da entidade empregado? s/n
Usuário: s.
Assistente: Indique o atributo
Usuário: registro.
Assistente: Existe mais algum atributo a ser definido? s/n
Usuário: n.

A partir das respostas do usuário, os seguintes fatos são gerados:

tipo(atributo,registro).
tipo(possui,empregado,registro).

Da mesma forma, os seguintes atributos e relações são gerados

1. Atributos
   - tipo(atributo,número).
   - tipo(atributo,quantidade).
   - tipo(atributo,característica).
   - tipo(atributo,informação).
   - tipo(atributo,horário).
   - tipo(atributo,marcar).
   - tipo(atributo,registro).
   - tipo(atributo,tipo).
   - tipo(atributo,formulário).
   - tipo(atributo,função).
   - tipo(atributo,horas).
   - tipo(atributo,partido).
   - tipo(atributo,trabalho).
   - tipo(atributo,localização).
   - tipo(atributo,data).
   - tipo(atributo,nome).

2. Relação entre atributos e entidades/relacionamento
   - tipo(possui,executado,trabalho).
   - tipo(possui,tipo de trabalho,característica).

- tipo(possui,possui,informação).
- tipo(possui,coletor de dado,horário).
- tipo(possui,utilizam,marcar).
- tipo(possui,trabalhar,tipo).
- tipo(possui,reportam,formulário).
- tipo(possui,filiam-se,data).
- tipo(possui,tem-1-1,registro).
- tipo(possui,atividade,tipo).
- tipo(possui,companhia,nome).
- tipo(possui,sede,número).
- tipo(possui,localizam-se,função).
- tipo(possui,estado,nome).
- tipo(possui,dividem-se,quantidade).
- tipo(possui,departamento,nome).
- tipo(possui,tem,localização).
- tipo(possui,relógio de ponto,data).
- tipo(possui,registra,horas).
- tipo(possui,horário,horas).
- tipo(possui,tem-1,quantidade).
- tipo(possui,representadas,quantidade).
- tipo(possui,união de trabalhador,partido).

6.2.3 Modelo Textual

O modelo textual gerado é uma listagem das entidades, relacionamentos e atributos surgidos da especificação. A geração do modelo consiste na obtenção dos fatos que se encontram na base de dados e apresentação ao usuário na seguinte forma:

- Entidade

  união de trabalhador "é do tipo" entidade.
  tipo de trabalho "é do tipo" entidade.
  atividade "é do tipo" entidade.
  relógio de ponto "é do tipo" entidade.
  horário "é do tipo" entidade.
  empregado "é do tipo" entidade.
  departamento "é do tipo" entidade.
  estado "é do tipo" entidade.

estação de trabalho "é do tipo" entidade.
coletor de dado "é do tipo" entidade.
companhia "é do tipo" entidade.
sede "é do tipo" entidade.

- Relacionamento

executado "é do tipo" relacionamento.
representadas "é do tipo" relacionamento.
filiam-se "é do tipo" relacionamento.
reportam "é do tipo" relacionamento.
utilizam "é do tipo" relacionamento.
trabalhar "é do tipo" relacionamento.
registra "é do tipo" relacionamento.
dividem-se "é do tipo" relacionamento.
divide-se "é do tipo" relacionamento.
localizam-se "é do tipo" relacionamento.
possui "é do tipo" relacionamento.
tem "é do tipo" relacionamento.
tem-1 "é do tipo" relacionamento.
tem-1-1 "é do tipo" relacionamento.

- Atributos

número "é do tipo" atributo.
quantidade "é do tipo" atributo.
característica "é do tipo" atributo.
informação "é do tipo" atributo.
horário "é do tipo" atributo.
marcar "é do tipo" atributo.
registro "é do tipo" atributo.
tipo "é do tipo" atributo.
formulário "é do tipo" atributo.
função "é do tipo" atributo.
horas "é do tipo" atributo.
partido "é do tipo" atributo.
trabalho "é do tipo" atributo.

localização "é do tipo" atributo.
data "é do tipo" atributo.
nome "é do tipo" atributo.

- Relação entre atributos e entidades/relacionamento

executado "possui o atributo" trabalho.
tipo de trabalho "possui o atributo" característica.
possui "possui o atributo" informação.
coletor de dado "possui o atributo" horário.
utilizam "possui o atributo" marcar.
empregado "possui o atributo" registro.
trabalhar "possui o atributo" tipo.
reportam "possui o atributo" formulário.
filiam-se "possui o atributo" data.
tem-1 "possui o atributo" quantidade.
atividade "possui o atributo" tipo.
companhia "possui o atributo" nome.
sede "possui o atributo" número.
tem-1-1 "possui o atributo" registro.
tem "possui o atributo" localização
localizam-se "possui o atributo" função.
estado "possui o atributo" nome.
dividem-se "possui o atributo" quantidade.
departamento "possui o atributo" nome.
relógio de ponto "possui o atributo" data.
registra "possui o atributo" horas.
horário "possui o atributo" horas.
representadas "possui o atributo" quantidade.
união de trabalhador "possui o atributo" partido.

- Relação

estação de trabalho "tem relação" 'N' com executado.
tipo de trabalho "tem relação" 1 com executado.
companhia "tem relação" 1 com representadas.
união de trabalhador "tem relação" 'N' com representadas.

união de trabalhador "tem relação" 1 com filiam-se.
empregado "tem relação" 'N' com filiam-se.
atividade "tem relação" 'N' com reportam.
empregado "tem relação" 'N' com reportam.
coletor de dado "tem relação" 1 com utilizam.
empregado "tem relação" 'N' com utilizam.
estação de trabalho "tem relação" 'N' com trabalhar.
empregado "tem relação" 1 com trabalhar.
relógio de ponto "tem relação" 1 com tem-1.
departamento "tem relação" 1 com tem-1.
horário "tem relação" 'N' com registra.
relógio de ponto "tem relação" 1 com registra.
empregado "tem relação" 'N' com tem-1-1.
estação de trabalho "tem relação" 'N' com dividem-se.
departamento "tem relação" 1 com dividem-se.
departamento "tem relação" 'N' com divide-se.
sede "tem relação" 1 com divide-se.
estado "tem relação" 1 com localizam-se.
sede "tem relação" 'N' com localizam-se.
coletor de dado "tem relação" 1 com possui.
estação de trabalho "tem relação" 1 com possui.
sede "tem relação" 'N' com tem.
companhia "tem relação" 1 com tem.
companhia "tem relação" 1 com tem-1-1.

- Par_ordenado

companhia "através de" tem "relaciona-se com" sede "com tipos" 1 e 'N'.
sede "através de" localizam-se "relaciona-se com" estado "com tipos" 'N' e 1.
sede "através de" divide-se "relaciona-se com" departamento "com tipos" 1 e 'N'.
companhia "através de" tem-1-1 "relaciona-se com" empregado "com tipos" 1 e 'N'.
relógio de ponto "através de" registra "relaciona-se com" horário "com tipos" 1 e 'N'.
empregado "através de" reportam "relaciona-se com" atividade "com tipos" 'N' e 'N'.
tipo de trabalho "através de" executado "relaciona-se com" estação de trabalho "com tipos" 1 e 'N'.
união de trabalhador "através de" representadas "relaciona-se com" companhia "com tipos" 'N' e 1.

empregado "através de" filiam-se "relaciona-se com" união de trabalhador "com tipos" 'N' e 1.

empregado "através de" utilizam "relaciona-se com" coletor de dado "com tipos" 'N' e 1.

empregado "através de" trabalhar "relaciona-se com" estação de trabalho "com tipos" 1 e 'N'.

departamento "através de" tem-1 "relaciona-se com" relógio de ponto "com tipos" 1 e 1.

departamento "através de" dividem-se "relaciona-se com" estação de trabalho "com tipos" 1 e 'N'.

estação de trabalho "através de" possui "relaciona-se com" coletor de dado "com tipos" 1 e 1.

6.3 Conclusão

Nosso objetivo, com este exemplo, foi ilustrar a abordagem de um problema utilizando o modelo de assistente criado. A utilização do mesmo exemplo estudado por Chen [Che 83a] tem como objetivo mostrar a implementação das regras estabelecidas por ele, por outros autores e por nós, observando a viabilidade de se obter um diagrama entidade relacionamento de forma automática a partir de sentenças em linguagem natural. Comparando o diagrama ER por nós obtido (Figura 6.10) com o diagrama obtido por Chen (Figura 6.11), percebemos diferenças ocasionadas pela tradução das sentenças. Estas diferenças são superficiais. O importante é que cada um dos diagramas corresponde a sua descrição em linguagem natural.

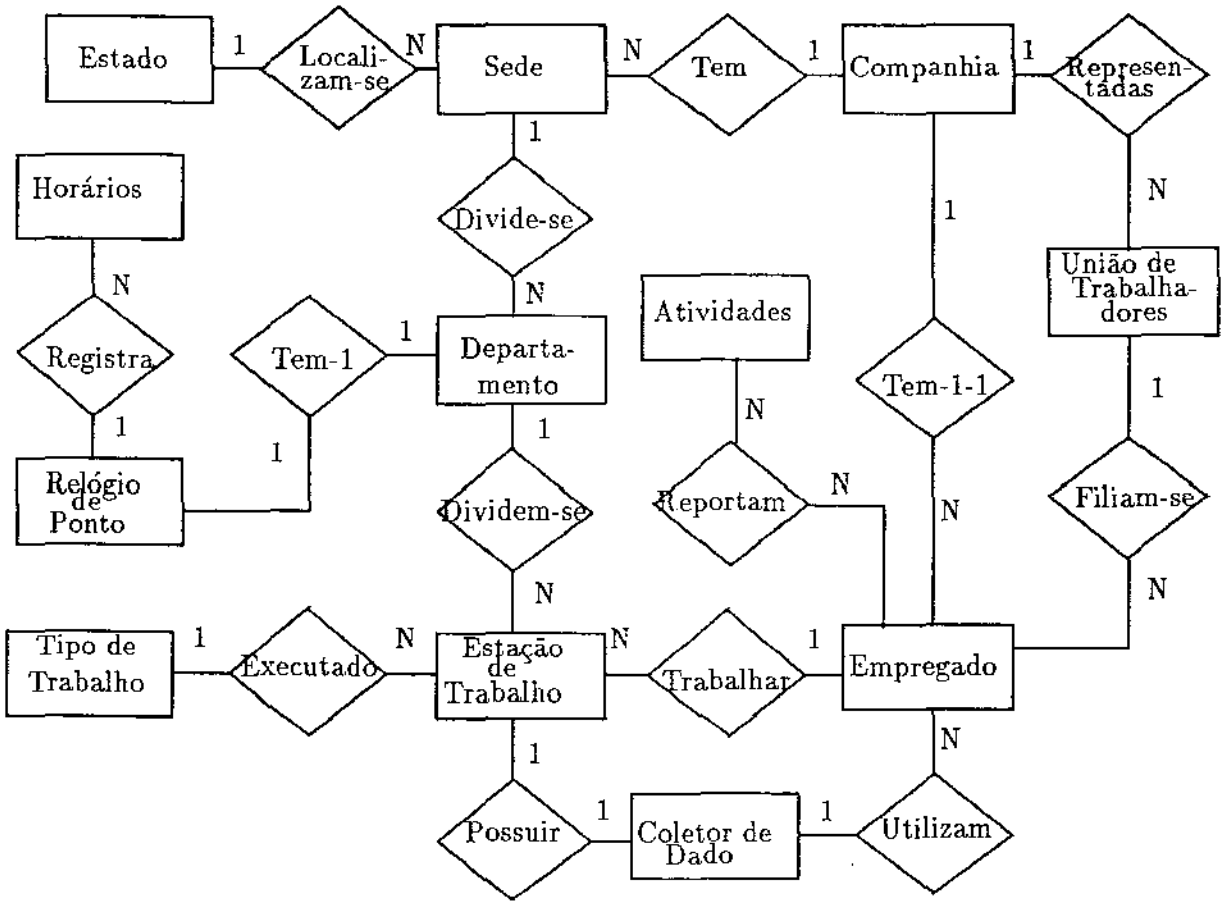


Figura 6.10: Diagrama ER Resultante da Verificação

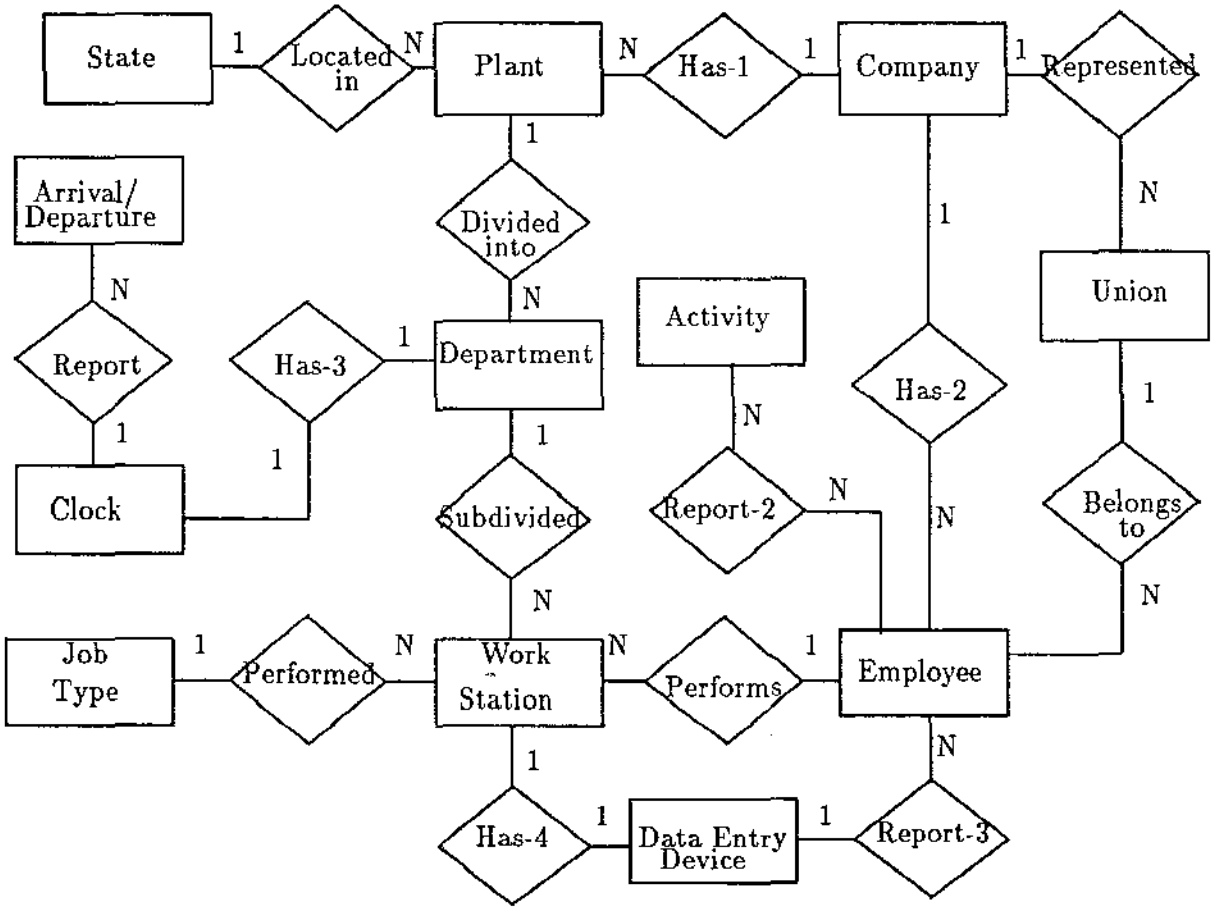


Figura 6.11: Diagrama ER Apresentado por Chen

Capítulo 7

Considerações Finais e Trabalhos Futuros

7.1 Considerações Finais

Nesse trabalho foi desenvolvido um assistente especialista em metodologias, que auxilia o projetista de *software* durante a fase de especificação de requisitos, considerada uma etapa crucial no processo de desenvolvimento.

A ferramenta consta das seguintes partes:

- Interface em linguagem natural;
- Base de dados;
- Máquina de inferência;
- Base de conhecimentos.

A interface é responsável pela interação da ferramenta com o usuário; essa comunicação é efetuada através de um diálogo em linguagem natural, dirigido pelo próprio assistente. As respostas obtidas através dessa interação são transformadas, com o auxílio de uma gramática DCG, em primitivas do MER, e armazenadas na base de dados. Utilizando a máquina de inferência, o assistente analisa essas primitivas e, consultando o conhecimento específico sobre o MER, armazenado na base de conhecimentos, verifica possíveis erros e/ou incompletude. Finalmente, o modelo textual correspondente a especificação semi-formal é gerado, a partir do conteúdo da base de dados.

Para que a ferramenta fosse desenvolvida, vários passos intermediários tiveram que ser percorridos:

- Definição de um método de obtenção da especificação informal: o método escolhido foi o questionário, instrumento de fácil entendimento e utilização e que facilita o processo, tornando-o rápido e permitindo a formalização das tarefas relativas à fase de análise;
- Definição de um formalismo gramatical que permitisse a análise sintática das respostas, dadas pelo usuário, em linguagem natural: o formalismo escolhido foram as DCGs, pois são fáceis de serem expandidas, além de serem abrangentes o suficiente para analisar as respostas recebidas pelo assistente;

- Escolha de um modelo para representação da aquisição: o modelo entidade-relacionamento foi eleito, visto ser um modelo amplamente utilizado e que permite uma clara estruturação das especificações;
- Escolha de um método para representação do conhecimento: as regras de produção foram escolhidas devido ao fato de que a sua sintaxe é condizente com a sintaxe das primitivas do MER;
- Extensão das regras heurísticas estabelecidas por Chen: para realizar a transformação de sentenças compostas, sentenças com preposições e sentenças com pronome relativo, para primitivas do MER. Também foi necessário a determinação de regras para estabelecer as cardinalidades de uma relação;
- Automação do processo de obtenção de requisitos: a especificação de requisitos é automática, enquanto que no trabalho apresentado por Loh ela é feita manualmente.

7.2 Trabalhos Futuros

Como mencionado no Capítulo 1, o processo de desenvolvimento de *software* consta de duas fases: abstração e concretização. O produto final desse trabalho é a obtenção de uma especificação semi-formal. A extensão natural da tese seria a obtenção de uma especificação formal, que serviria como base para o processo de concretização.

Com respeito ao modelo de representação, o MER poderia ser substituído por novos modelos, se isso fosse desejável. Nesse caso, alterações na base de conhecimentos deveriam ser efetuadas de forma a retratar o conhecimento sobre o modelo escolhido. Além disso, comparações entre os diferentes modelos poderiam ser realizadas.

A fonte das regras utilizadas na base de conhecimentos foram os diversos trabalhos sobre o assunto. Entretanto, novas regras poderiam ser incorporadas ao sistema através da ampla utilização da ferramenta e da observação do seu comportamento.

Finalmente, com relação à linguagem natural, um tratamento de sinônimos e ambigüidade seria extremamente útil, visto que dessa maneira o número de entidades e relacionamentos utilizados seria bastante reduzido.

Bibliografia

[Ari 87] Arity Corp., *Using the Arity/Prolog Interpreter and Compiler*, Manual de Usuário, Arity Corporation, 1987.

[Ari 92a] Arias C. & Carvalho A., Um Modelo de Assistente para Especificação de Requisitos, *Relatório Técnico 08/92*, Departamento de Ciência da Computação, UNICAMP, 1992.

[Ari 92b] Arias C. & Carvalho A., Um Modelo para Coleta de Informação, *Relatório de Pesquisa 03/92*, Departamento de Ciência da Computação, UNICAMP, 1992.

[Ari 92c] Arias C. & Carvalho A., Um Assistente Especialista para Especificação de Requisitos, *Submetido para VI Simpósio Brasileiro de Engenharia de Software.*

[Ari 92d] Arias C. & Carvalho A., Un Asistente Experto para Especificación de Requisitos según el Modelo Entidad Relacionamiento, *Submetido para VII Conferencia Científica de Ingeniería y Arquitectura*, Cuba.

[Ari 92e] Arias C. & Carvalho A., *FER: Uma Ferramenta para Especificação de Requisitos*, Manual de Usuário, Departamento de Ciência da Computação, UNICAMP, 1992.

[Bal 85] Balzer R., A 15 Year Perspective on Automatic Programming, *IEEE Transactions on Software Engineering*, Vol. SE-11, No. 11, Pg. 1257-1268, 1985.

[Bar 81] Barr A. & Feigenbaum E., *The HandBook of Artificial Intelligence*, Volumes I e II, Addison-Wesley Inc., 1981.

[Bor 87] Boria J., *Ingeniería de Software*, Editorial Kapelusz S. A., 1987.

[Bow 87] Bowen D., Byrd L., Chung P., Pereira F., Pereira L., Rae R. & Warren D., *Edinburgh Prolog, The New Implementation User's Manual*, AI Applications Institute, University of Edinburgh, 1987.

[Bru 87] Bruck Rotenberg H., Programação Orientada a Objetos: um Enfoque de Engenharia de Software, *Tese de Mestrado*, PUC-Rio de Janeiro, 1987.

[Car 86] Carnota R. & Goldszein M., *Inteligencia Artificial: Lógica y Representación del Conocimiento*, Campinas Editora UNICAMP, 1986.

[Car 88] Carnota R. & Teszkiewicz A., *Sistemas Expertos y Representación del Conocimiento*, Curitiba, EBAI, 1988.

[Car 89] Carvalho A., Logic Grammars and Pronominal Anaphora, *Tese de Doutorado*, Departamento de Ciência da Computação, Universidade de Reading, Inglaterra, 1989.

[Cas 88] Castilho J., *Especificações Formais*, EBAI, 1988.

[Che 76] Chen P., The Entity-Relationship Model-Towards a Unified View of Data, *ACM-TODS*, Vol. 1, No. 1, Pg. 9-36, 1976.

[Che 83a] Chen P., English Sentence Structure and Entity-Relationship Diagrams, *Information Sciences*, No 29, Pg. 127-149, 1983.

[Che 83b] Chen P., Chung I. & Nakamura I., A Decomposition of Relations Using the Entity-Relationship Approach, *Entity-Relationship Approach to Information Modeling and Analysis*, P.P. Chen (ed.), Elsevier Science Publishers B.V., (North-Holland), Pg. 149-171, 1983.

[Clo 84] Clocksin W. & Mellish C., Programming in Prolog, Segunda Edição, Springer-Verlag, Berlin, Heidelberg, 1984.

[Dah 83] Dahl V. & McCord M., Treating Coordination in Logic Grammars, *American Journal of Computational Linguistics*, Vol 9, No 2, Pg. 69-91, 1983.

[Doc 89] Docker T. & France R., Flexibility and Rigour in Structured Analysis, *Information Processing*, G. X. Ritter (ed), Elservier Science Publishers B. V. (North-Holland), Pg. 89-94, 1989.

[Dol 88] Dolder H. & Lubomirsky E. *Sistema Experto para el Diseño de Bases de Datos*, Editorial Kapelusz, Buenos Aires, 1988.

[Dub 83] Dubois J., Giacomo M., Guespin L., Marcellesi C., Marcellesi J. & Mevel J., *Dicionário de Lingüística*, Editora Cultrix, São Paulo, 1983.

[Hae 88] Haeberer A., Veloso P. & Baum G., *Formalización del Proceso de Desarrollo de Software*, EBAI, Editorial Kapelusz, 1988.

[Har 88] Harmon P. & King D., *Sistemas Especialistas*, Editora Campus, 1988.

[Har 82] Hartman W., Matthers H. & Proeme A., *Manual de los Sistemas de Información*, Editorial Paraninfo S.A., España, 1982.

[Heu 88] Heuser C., *Modelagem Conceitual de Sistemas*, EBAI, Editorial Kapelusz, 1988.

[Hsu 88] Hsu C., Perry A., Bouziane M. & Cheung W., TSER: A Data Modeling System Using the Two-Stage Entity-Relationship Approach, *Entity-Relationship Approach*, ST. March (ed.), Elsevier Science Publishers B.V., (North-Holland), Pg. 497-514, 1988.

[Jac 83] Jackson M., *System Development*, Prentice-Hall, 1983.

[Ken 83] Kent W., Fact-Based Data Analysis and Design, *Entity-Relationship Approach to Software Engineering* , Elsevier Science Publishers B.V., (North-Holland), Pg. 3-53, 1983.

[Koz 88] Kozaczynski W. & Lilien L., An Extended Entity-Relationship Database Specification and its Automatic Verification and Transformation into the Logical Relational Design, *Entity-Relationship Approach*, ST. March (ed.), Elsevier Science Publishers B.V., (North-Holland), Pg. 533-549, 1988.

[Kvi 88] Kvitca A., *Resolución de Problemas con Inteligencia Artificial*, Curitiba, EBAI, 1988.

[Leh 84] Lehman M., A Further Model of Coherent Programming Processes, *IEEE Proceedings of Software Process Workshop*, IEEE Comp. Soc., pg 27-31, 1984.

[Len 82] Lenat D., The Nature of Heuristics, *Artificial Intelligence*, North-Holland Publishing Company, No. 2, Pg. 189-249, 1982.

[Lin 83] Lindgreen P., Entity Sets and their Description, *Entity-Relationship Approach to Software Engineering*, Davis C., Jajodia S., Ann-Beng P., Yeh R.(eds.), Elsevier Science Publishers B.V., (North-Holland), Pg. 91-110, 1983.

[Lin 87] Lindgreen P., Entity from a Systems Point of View, *Entity-Relationship Approach*, Spaccapietra S.(ed.), Elsevier Science Publishers B.V., (North-Holland), Pg. 119-131, 1987.

[Lin 88] Ling T., A Three Level Schema Architecture ER-based Data Base Management System, *Entity-Relationship Approach*, ST. March (ed.), Elsevier Science Publishers B.V., (North-Holland), Pg. 205-213, 1988.

[Loh 88] Loh S., Método de Coleta e Documentação de Dados para a Modelagem de Sistemas com Estudos de Casos, *Trabalho de Conclusão*, Universidade Federal do Rio Grande do Sul, 1988.

[Luc 87] Lucena C., *Inteligência Artificial e Engenharia de Software*, Jorge Zahar Editor, PUC-RJ, IBM Brasil, 1987.

[Mal 87] Malpas J., *Prolog: A Relational Language and its Applications*, Prentice-Hall, 1987.

[Man 87] Mannino M., Choobineh J. & Hwang J., Acquisition and use of Contextual Knowledge in a Form-Driven Data Base Design Methodology, *Entity-Relationship Approach* , Spaccapietra S.(ed.), Elsevier Science Publishers B.V., (North-Holland), Pg. 361-372, 1987.

[Men 88] Mendes S. & Aguiar T., *Métodos para especificação de sistemas*, EBAI, 1988.

[Mir 91] Miriyala K. & Harandi M., Automatic Derivation of Formal Software Specifications from Informal Descriptions, *IEEE Transactions on Software Engineering*, Vol. 17, No 10, Pg. 1126-1142, 1991.

[Mor 86] Moriconi M., A Designer/Verifier's Assistant, *IEEE Transactions on Software Engineering*, Vol. 5, No 4, Pg. 387-401, 1979.

[Nad 71] Nadler G., *Diseño de Sistemas de Producción* , Editorial El Ateneo, 1971.

[Naz 91] Nazzetta R., Um Sistema Baseado em Conhecimento para Configuração e Supervisão de Algoritmos de Contrôle Adaptativo, *Tese de Mestrado*, FEE Departamento de Computação e Automação Industrial, Fevereiro, 1991.

[Nis 86] Niskier C., PRISMA: Utilização de Paradigmas Complementares para a Aquisição da Especificação de Software, *Tese de Mestrado*, Departamento de Informática, PUC/RJ, 1986.

[Per 80] Pereira F. & Warren D., Definite Clause Grammars for Language Analysis, *Artificial Intelligence*, Vol. 13, Pg. 231-278, 1980.

[Pes 86] Pessoa T., Projeto Assistido por Computador: Um Sistema Especialista para o Método de Jackson, *Tese de Mestrado*, Departamento de Informâtica, PUC/RJ, 1986.

[Pra 90] Prager J., Lamberti D., Gardner D. & Balzac S., Reason: An Intelligent User Assistant for Interative Environments, *IBM Systems Journal*, Vol. 29, No. 1, Pg. 141-164, 1990.

[Pre 87] Pressman R., *Software Engineering a Practitioner's Approach*, McGraw-Hill, segunda edição, 1987.

[Reu 91] Reubenstein H. & Waters R., The Requirements Apprentice: Automated Assistance for Requirements Acquisition, *IEEE Transactions on Software Engineering*, Vol. 17, No 3, Pg. 226-240, 1991.

[Sav 88] Savadovsky P., *A Construção de Interpretadores para Linguagem Natural*, EBAI, 1988.

[Sch 88] Schoen E. & Smith R., Design of Knowledge-Based Systems with a Knowledge-Based Assistant, *IEEE Transactions on Software Engineering*, Vol. 14, No 12, Pg. 1771-1789, 1988.

[Sco 89] Scott D. & Souza C., *Conciliatory Planning for Extended Descriptive Texts*, PUC, Rio de Janeiro, Monografias em Ciência da Computação, Número 13, Editor Paulo Veloso, 1989.

[Sen 87] Senn J., *Analisis y diseño de Sistemas de Información*, McGraw-Hill, 1987.

[Set 86] Setzer V., *Projeto Lógico e Projeto Físico de Banco de Dados*, V Escola de Computação , Belo Horizonte, 1986.

[Tak 90] Takahashi T. & Liesenberg H., *Programação Orientada a Objetos* , VII Escola de Computação, São Paulo, 1990.

[Wat 85] Waters R., The Programmer's Apprentice a Session with KBEmacs, *IEEE Transactions on software Engineering*, Vol. SE-11, No 11, Pg. , 1985.

[Wil 87] Williams D., *Análise e Projeto Estruturado de Sistemas*, Livros Técnicos e Científicos Editora, 1987.

Apêndice A

Base de Conhecimentos

Apresentaremos aqui as regras da Base de Conhecimentos, indicando o tipo de "conhecimento heurístico" que a base contém. A palavra "heurística" indica o conhecimento capaz de sugerir ações plausíveis a serem seguidas ou ações não plausíveis a serem evitadas [Len 82]. Estas regras foram obtidas pesquisando-se a bibliografia e os exemplos de utilização do modelo entidade-relacionamento ([Che 76];[Nis 86];[Lin 88];[Koz 88]).

1. Regras de Modificação

   . regra1 :

   Se duas entidades possuem cardinalidade N e M com um relacionamento
   então o relacionamento é do tipo "muitos para muitos"

   . regra2 :

   Se duas entidades possuem cardinalidade 1 e N com um relacionamento
   então o relacionamento é do tipo "um para muitos"

   . regra3 :

   Se duas entidades possuem cardinalidade 1 com um relacionamento
   então o relacionamento é do tipo "um para um"

   . regra4 :

   Se duas entidades possuem cardinalidade N e 1 com um relacionamento
   então o relacionamento é do tipo "muitos para um"

   . regra5 :

   Se três entidades relacionam-se com o mesmo relacionamento
   então o relacionamento é do tipo "ternário"

   . regra6 :

   Se três entidades possuem cardinalidade 1 com um relacionamento
   então o relacionamento é do tipo "ternário um para um"

   . regra7 :

   Se três entidades possuem cardinalidade N com um relacionamento

então o relacionamento é do tipo "ternário muitos para muitos"

. regra8 :

Se existe um relacionamento que interage com uma só entidade
então define-se uma entidade fictícia "sem nome", com cardinalidade default 1.

. regra9 :

Se um relacionamento interage com quatro entidades
então defina um novo relacionamento entre duas entidades

2. Regras de Erros

. regra1 :

Se existe um relacionamento que não se relaciona com nenhuma entidade
então erro: defina uma entidade para ele

. regra2 :

Se existe uma entidade que não se relaciona com nada
então erro: defina um relacionamento para ela

. regra3 :

Se existe uma entidade que não possui atributos
então erro: defina atributos para ela

. regra4 :

Se existe um relacionamento que não possui atributos
então erro: defina atributos para ele

. regra5 :

Se existe um atributo que não está ligado com nenhuma entidade ou relacionamento
então erro: defina uma entidade ou relacionamento para ele

. regra6 :

Se existe uma entidade definida como "sem nome"
então defina o nome da entidade e o relacionamento do qual ela faz parte

. regra7 :

Se existem dois relacionamentos interagindo entre si
então erro: redefina o relacionamento

. regra8 :

Se um relacionamento interage com quatro entidades
então erro: o relacionamento pode, no máximo, ser ternário

3. Regras de Informação

. regra1 :

Se o tipo de representação utilizada é MER

então defina os objetos sobre os quais se deseja guardar informação

. regra2 :

Se um conceito é uma entidade

então ele é um objeto da realidade sobre o qual se deseja guardar informação

. regra3 :

Se um conceito é um relacionamento

então ele é uma interação entre dois ou três objetos

. regra4 :

Se um conceito é um atributo

então ele é uma característica ou propriedade do objeto

. regra5 :

Se uma entidade tem cardinalidade 1

então uma instância de uma entidade interage com um relacionamento

. regra6 :

Se uma entidade possui "papéis"

então estes são as funções que a entidade desempenha no relacionamento

. regra7 :

Se duas entidades possuem um relacionamento 1 para N (do tipo "um para muitos")

então uma instância de uma entidade interage com várias instâncias da outra entidade.

. regra8 :

Se duas entidades possuem um relacionamento 1 para 1 (do tipo "um para um")

então uma instância de uma entidade interage com uma instância da outra entidade.

. regra9 :

Se duas entidades possuem um relacionamento N para 1 (do tipo "muitos para um")

então várias instâncias de uma entidade interagem com uma instância da outra entidade.

. regra10 :

Se duas entidades possuem um relacionamento N para M (do tipo "muitos para muitos")

então várias instâncias de uma entidade interagem com várias instâncias da outra entidade.

. regra11 :

Se um relacionamento é do tipo ternário

então três entidades compartilham o mesmo relacionamento

Apêndice B

Outro Exemplo de Utilização

Apresentaremos neste apêndice outro exemplo de utilização do assistente, dessa vez para um consultório médico, mostrando a interpretação de algumas sentenças. Este exemplo é semelhante ao apresentado por Loh [Loh 88], e mostra como obter a especificação de requisitos de uma maneira mais rápida que a proposta por ele.

B.1 Especificação

1. A organização.

   (a) Indicar de que é composta a organização.
   Resposta:

   O consultório tem secretária.

   (b) Indicar o objetivo da organização.

   (c) Indicar os fornecedores da organização e o que fornecem.

   (d) Citar outros orgãos externos à organização com os quais ela possui algum relacionamento.
   Resposta:

   Outras instituições relacionam-se com os pacientes.

   (e) Indicar que relatórios ou informações precisam ser passados para algum orgão externo.
   Resposta:

   O consultório emite a carta do encaminhamento do paciente.

2. Descrição funcional.

   (a) Indicar a atividade básica da organização.
   Resposta:

   Os pacientes marcam consultas.

   (b) Com relação a atividade principal da organização, indicar os dados que são utilizados ou produzidos durante essa atividade.

(c) Descrever informalmente como cada atividade (ou função) é realizada, e que parte da organização a realiza.
Respostas:

O setor de consultas mantém os exames, mantém a história da doença e mantém os antecedentes da família.

A secretária controla o pagamento e mantém um cadastro de doenças.

O consultório mantém o histórico do paciente.

3. Características estruturais da organização.

(a) Indicar as partes que compõem a organização.
Resposta:

A secretária atende o paciente, cuida do consultório, mantém o histórico do paciente e paga contas.

(b) Indicar os objetivos de cada parte.

(c) Se alguma parte da organização se relacionar com outra, indicar com quem e através de que atividade.

4. Clientes/Usuários da organização.

(a) Indicar quais os benefícios que os usuários recebem da organização.
Respostas:

Os pacientes recebem atendimento do médico.

O atendimento do médico é rápido, agradável e correto.

O pagamento é realizado com cheque.

O consultório é bonito.

(b) Indicar com que funções interagem os usuários.

(c) Dar sugestões em termos de atividade que os usuários gostariam de ter.

5. Fluxo de informação.

(a) Para cada atividade fundamental, descrever que dados utiliza, possui ou emite.
Respostas:

O setor de consultas têm o registro de dados e têm o registro de contas.

O setor de consultas têm a história do progresso do paciente.

(b) Indicar documentos, informes ou outras saídas, e a que setor (ou parte da organização) pertencem.
Resposta:

O relatório do caixa pertence à secretária.

(c) Indicar documentos e informes que pertencem a algum orgão externo.

(d) Propor uma atividade e indicar que informações ela requer para ser realizada.

(e) Listar os novos elementos de dados que os usuários gostariam de ver no novo sistema. Indicar também de onde surgem e onde são utilizados.

6. O futuro da organização.

(a) Indicar, segundo o seu critério, que novas seções deveriam existir dentro da organização, especificando com quem deveriam se relacionar.
Respostas:

O consultório precisa de uma lanchonete.

A secretária com os médicos comeriam na lanchonete.

(b) Indicar o que fazem as novas seções apresentadas acima.
Resposta:

A lanchonete atenderia aos pacientes.

(c) Indicar que informações ou dados podem ser acrescentados a alguma atividade já descrita ou a uma nova atividade, para melhorar o seu funcionamento.
Resposta:

O setor de consultas manteria um cadastro de doenças.

(d) Indicar que novo serviço poderia ser prestado pela organização, dizendo que função executaria.

B.1.1 Interpretação das Sentenças

Veremos os diagramas entidade-relacionamento de algumas sentenças em particular. Na Figura B.7 será apresentado o diagrama completo.

A sentença 1:

1. O consultório tem secretária

produz a seguinte interpretação:

E1= consultório

E2= secretária

Tipo1= 1

Tipo2= 1

Rel= tem

e gera as seguintes primitivas:

tipo(entidade,consultório).

tipo(entidade,secretária).

tipo(relacionamento,tem).

tipo(relação,secretária,1,tem).

tipo(relação,consultório,1,tem).

tipo(par_ordenado,consultório,tem,secretária,1,1).

A sentença 2:

2. Outras instituições relacionam-se com os pacientes

produz a seguinte interpretação:

E1= instituição

E2= paciente

Tipo1= 1

Tipo2= N

Rel= relacionam-se

e gera as seguintes primitivas:

tipo(entidade,paciente).

tipo(entidade,instituição).

tipo(relacionamento,relacionam-se).

tipo(relação,paciente,'N',relacionam-se).

tipo(relação,instituição,1,relacionam-se).

tipo(par_ordenado,instituição,relacionam-se,paciente,1,'N').

O diagrama ER correspondente à interpretação dessa sentença é mostrado na Figura B.1.

A sentença 3:

3. O consultório emite a carta do encaminhamento do paciente.

produz a seguinte interpretação:

E1= consultório

E2= carta de encaminhamento de paciente

Tipo1= 1

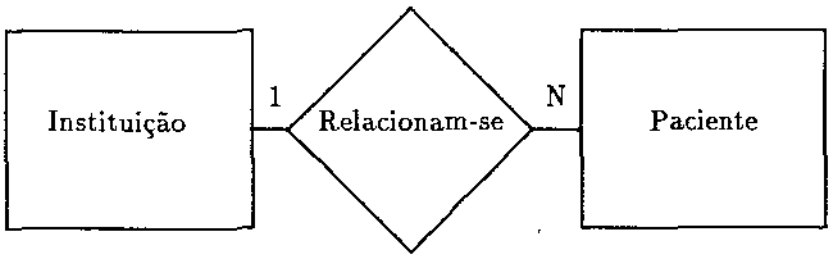


Figura B.1: Diagrama ER da Sentença 2.

Tipo2= 1

Rel= emite

e gera as seguintes primitivas:

tipo(entidade,consultório)

tipo(entidade,carta de encaminhamento de paciente).

tipo(relacionamento,emite).

tipo(relação,consultório,1,emite).

tipo(relação,carta de encaminhamento de paciente,1,emite).

tipo(par_ordenado,consultório,emite,carta de encaminhamento de paciente,1,1).

A sentença 4:

4. Os pacientes marcam consultas

produz a seguinte interpretação:

E1= paciente

E2= consulta

Tipo1= N

Tipo2= 1

Rel= marcam

e gera as seguintes primitivas:

tipo(entidade,paciente).

tipo(entidade,consulta).

tipo(relacionamento,marcam).

tipo(relação,consulta,1,marcam).

tipo(relação,paciente,'N',marcam).

tipo(par_ordenado,paciente,marcam,consulta,'N',1).

As sentenças 5, 6 e 7:

5. O setor de consultas mantém os exames, mantém a história da doença e mantém os antecedentes da família

6. A secretária controla o pagamento e mantém um cadastro de doenças

7. O consultório mantém o histórico do paciente

produzem a seguinte interpretação:

E1= setor de consulta

E2= exame&(história de doença)&antecedente de família

Tipo1= 1

Tipo2= N

Rel= mantém&mantém&mantém

E1= secretária

E2= pagamento&cadastro de doença

Tipo1= 1

Tipo2= 1

Rel= controla&mantém

E1= consultório

E2= histórico de paciente

Tipo1= 1

Tipo2= 1

Rel= mantém

e geram as seguintes primitivas:

tipo(entidade,secretária).

tipo(entidade,setor de consulta).

tipo(entidade,consultório).

tipo(entidade,pagamento).

tipo(relacionamento,controla).

tipo(relação,secretária,1,controla).

tipo(relação,pagamento,1,controla).

tipo(par_ordenado,secretária,controla,pagamento,1,1).

tipo(atributo,antecedente de família).

tipo(atributo,história de doença).

tipo(atributo,exame).

tipo(atributo,cadastro de doença).

tipo(atributo,histórico de paciente).

tipo(possui,setor de consulta,antecedente de família).

tipo(possui,setor de consulta,história de doença).

tipo(possui,setor de_consulta,exame).

tipo(possui,secretária,cadastro de doença).

tipo(possui,consultório,histórico de paciente).

O diagrama correspondente à interpretação das sentenças 5, 6 e 7 é mostrado na Figura B.2. O verbo "manter" estabelece uma relação do tipo atributo entre uma entidade e uma característica do mesmo. Certos autores, como Loh, também consideram os verbos "ter" e "possuir" como pertencendo a essa mesma classe. Na nossa implementação, só o verbo "manter" é classificado dentro dessa categoria.

A sentença 8:

8. A secretária atende o paciente, cuida do consultório, mantém o histórico do paciente e paga contas

produz a seguinte interpretação:

E1= secretária

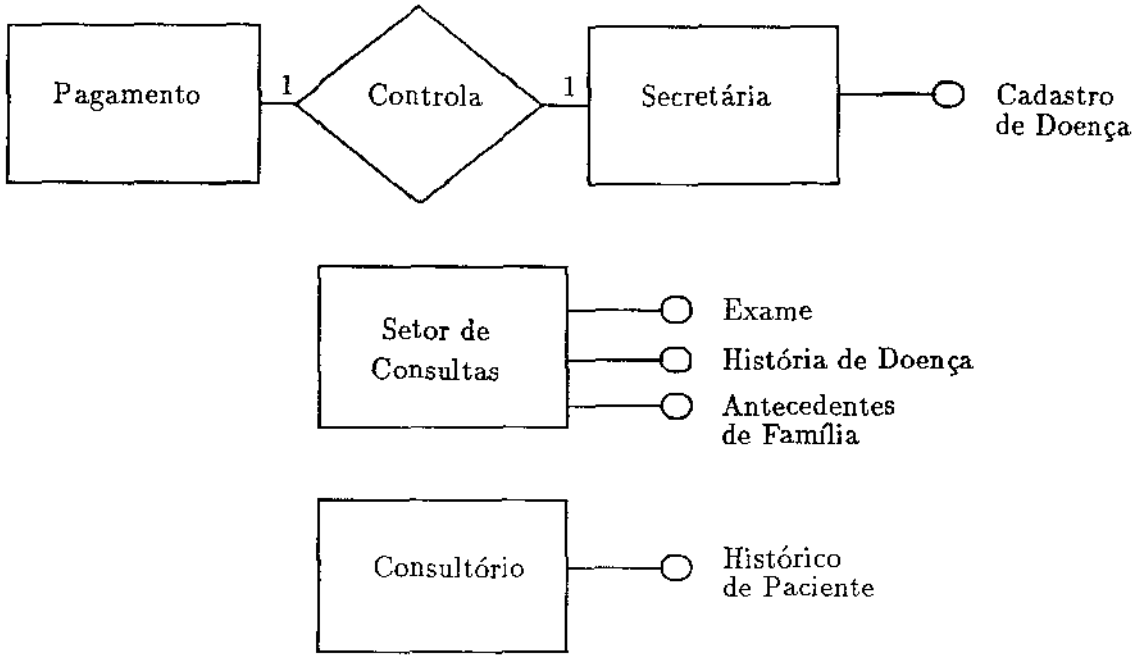


Figura B.2: Diagrama ER das Sentenças 5, 6 e 7

E2= paciente&consultório&(histórico de paciente)&conta

Tipo1= 1

Tipo2= 1

Rel= atende&cuida&mantém&paga

e gera as seguintes primitivas:

tipo(entidade,secretária).

tipo(entidade,paciente).

tipo(entidade,conta).

tipo(entidade,consultório).

tipo(atributo,histórico de paciente).

tipo(relacionamento,paga).

tipo(relacionamento,cuida).

tipo(relacionamento,atende).

tipo(possui,secretária,histórico de paciente).

tipo(relação,conta,1,paga).

tipo(relação,secretária,1,paga).

tipo(relação,secretária,1,cuida).

tipo(relação,consultório,1,cuida).

tipo(relação,secretária,1,atende).

tipo(relação,paciente,1,atende).

tipo(par_ordenado,secretária,paga,conta,1,1).

tipo(par_ordenado,secretária,cuida,consultório,1,1).

tipo(par_ordenado,secretária,atende,paciente,1,1).

A sentença 9:

9. Os pacientes recebem atendimento do médico

produz a seguinte interpretação:

E1= paciente

E2= atendimento de médico

Tipo1= N

Tipo2= 1

Rel= recebem

e gera as seguintes primitivas:

tipo(entidade,paciente).

tipo(entidade,atendimento de médico).

tipo(relacionamento,recebem).

tipo(relação,atendimento de médico,1,recebem).

tipo(relação,paciente,'N',recebem).

tipo(par_ordenado,paciente,recebem,atendimento de médico,'N',1).

A sentença 10:

10. O atendimento do médico é rápido, agradável e correto

produz a seguinte interpretação:

E1= atendimento de médico

E2= rápido&agradável&correto

Tipo1= 1

Tipo2= 1

Rel= é

e gera as seguintes primitivas:

tipo(entidade,atendimento de médico).

tipo(atributo,correto).

tipo(atributo,agradável).

tipo(atributo,rápido).

tipo(possui,atendimento de médico,correto).

tipo(possui,atendimento de médico,agradável).

tipo(possui,atendimento de médico,rápido).

Para a sentença 10 obtemos o diagrama apresentado na Figura B.3.

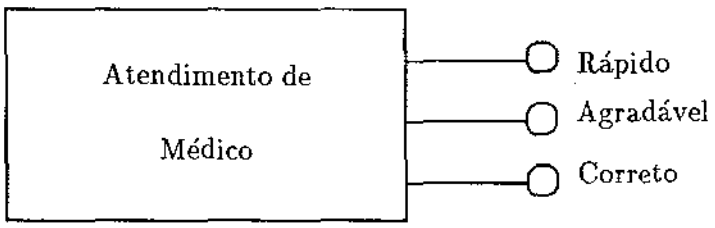


Figura B.3: Diagrama ER da Sentença 10.

A sentença

11. O pagamento é realizado com cheque

produz a seguinte interpretação:

E1= pagamento

E2= cheque

Tipo1= 1

Tipo2= 1

Rel= realizado

e gera as seguintes primitivas:

tipo(entidade,pagamento).

tipo(entidade,cheque).

tipo(relacionamento,realizado).

tipo(relação,cheque,1,realizado).

tipo(relação,pagamento,1,realizado).

tipo(par_ordenado,pagamento,realizado,cheque,1,1).

Da sentença 11 obtemos o diagrama apresentado na Figura B.4.

A sentença 12:

12. O consultório é bonito

produz a seguinte interpretação:

E1= consultório

E2= bonito

Tipo1= 1

Tipo2= 1

Rel= é

e gera as seguintes primitivas:

tipo(entidade,consultório).

tipo(atributo,bonito).

tipo(possui,consultório,bonito).

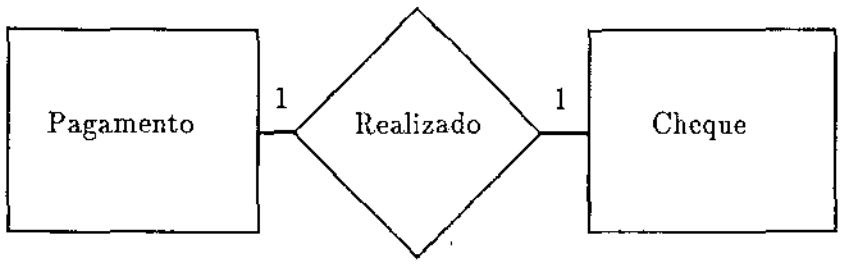


Figura B.4: Diagrama ER da Sentença 11.

Da sentença 12 obtemos o diagrama apresentado na Figura B.5.

As sentenças:

13. O setor de consultas têm o registro de dados e têm o registro de contas
14. O setor de consultas têm a história do progresso do paciente

produzem a seguinte interpretação:

E1= setor de consulta
E2= (registro de dado)&registro de conta
Tipo1= 1
Tipo2= 1
Rel= tem&tem

E1= setor de consulta
E2= história de progresso de paciente
Tipo1= 1
Tipo2= 1
Rel= tem

e geram as seguintes primitivas:

tipo(entidade,setor de consulta).

tipo(entidade,história de progresso de paciente).

tipo(entidade,registro de conta).

tipo(entidade,registro de dado).

Figura B.5: Diagrama ER da Sentença 12.

tipo(relação,história de progresso de paciente,1,tem).

tipo(relação,registro de conta,1,tem).

tipo(relação,setor de consulta,1,tem).

tipo(relação,registro de dado,1,tem).

tipo(par_ordenado,setor de consulta,tem,história de progresso de paciente,1,1).

tipo(par_ordenado,setor de consulta,tem,registro de conta,1,1).

tipo(par_ordenado,setor de consulta,tem,registro de dado,1,1).

A sentença

15. O relatório do caixa pertence à secretária

produz a seguinte interpretação:

E1= relatório de caixa

E2= secretária

Tipo1= 1

Tipo2= 1

Rel= pertence

e gera as seguintes primitivas:

tipo(entidade,secretária).

tipo(entidade,relatório de caixa).

tipo(relacionamento,pertence).

tipo(relação,secretária,1,pertence).

tipo(relação,relatório de caixa,1,pertence).

tipo(par_ordenado,relatório de caixa,pertence,secretária,1,1).

As sentenças 16 e 17:

16. O consultório precisa de uma lanchonete
17. A secretária com os médicos comeriam na lanchonete

produzem a seguinte interpretação:

E1= consultório
E2= lanchonete
Tipo1= 1
Tipo2= 1
Rel= precisa

E1= secretária&médico
E2= lanchonete
Tipo1= 1
Tipo2= 1
Rel= comeriam

e geram as seguintes primitivas:

tipo(entidade,consultório).

tipo(entidade,secretária).

tipo(entidade,médico).

tipo(entidade,lanchonete).

tipo(relacionamento,comeriam).

tipo(relacionamento,precisa).

tipo(relação,lanchonete,1,precisa).

tipo(relação,consultório,1,precisa).

tipo(relação,médico,1,comeriam).

tipo(relação,secretária,1,comeriam).

tipo(relação,lanchonete,1,comeriam).

tipo(par_ordenado,consultório,precisa,lanchonete,1,1).

tipo(par_ordenado,médico,comeriam,lanchonete,1,1).

tipo(par_ordenado,secretária,comeriam,lanchonete,1,1).

Na Figura B.6 vemos o diagrama correspondente à interpretação das sentenças 16 e 17.

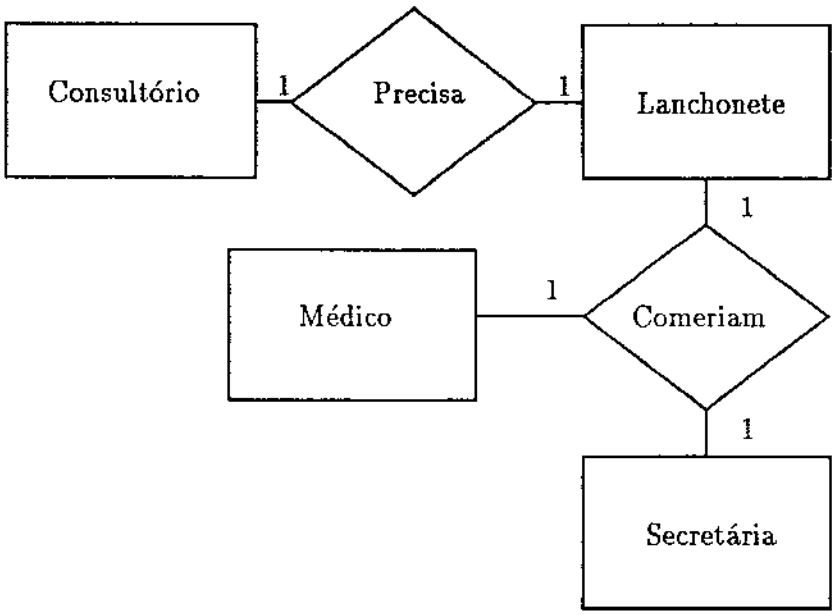


Figura B.6: Diagrama ER das Sentenças 16 e 17.

As sentenças 18 e 19:

18. A lanchonete atenderia aos pacientes

19. O setor de consultas manteria um cadastro de doenças

produzem a seguinte interpretação:

E1= lanchonete

E2= paciente

Tipo1= 1

Tipo2= N

Rel= atenderia

E1= setor de consulta

E2= cadastro de doença

Tipo1= 1

Tipo2= 1

Rel= manteria

e geram as seguintes primitivas:

tipo(entidade,lanchonete).

tipo(entidade,paciente).

tipo(entidade,setor de consulta).

tipo(atributo,cadastro de doença).

tipo(possui,setor de consulta,cadastro de doença).

tipo(relacionamento,atenderia).

tipo(relação,paciente,'N',atenderia).

tipo(relação,lanchonete,1,atenderia).

tipo(par_ordenado,lanchonete,atenderia,paciente,1,'N').

B.2 Verificação

A cardinalidade das seguintes relações foi alterada:

- tipo(par_ordenado,paciente,marcam,consulta,'N','N').
- tipo(par_ordenado,secretária,atende,paciente,1,'N').
- tipo(par_ordenado,instituição,relacionam-se,paciente,'N','N').
- tipo(par_ordenado,secretária,controla,pagamento,1,'N').
- tipo(par_ordenado,médico,comeriam,lanchonete,'N',1).
- tipo(par_ordenado,setor de consulta,tem,história de progresso de paciente,1,'N').
- tipo(par_ordenado,paciente,recebem,atendimento de médico,'N','N').
- tipo(par_ordenado,consultório,emite,carta de encaminhamento de paciente,1,'N').

Na consulta à base de conhecimentos foram disparadas as seguintes regras:

1. regra :

   Se duas entidades possuem cardinalidade N e M com um relacionamento
   então o relacionamento é do tipo "muitos para muitos"

2. regra :

   Se duas entidades possuem cardinalidade 1 e N com um relacionamento
   então o relacionamento é do tipo "um para muitos"

3. regra :

   Se duas entidades possuem cardinalidade 1 com um relacionamento
   então o relacionamento é do tipo "um para um"

4. regra :

   Se duas entidades possuem cardinalidade N e 1 com um relacionamento
   então o relacionamento é do tipo "muitos para um"

5. regra :

   Se existe uma entidade que não possui atributos
   então erro: defina atributos para ela

6. regra :

   Se existe um relacionamento que não possui atributos
   então erro: defina atributos para ela

7. regra :

   Se um conceito é um atributo
   então ele é uma característica ou propriedade do objeto

8. regra :

   Se um relacionamento interage com quatro entidades
   então erro: o relacionamento pode, no máximo, ser ternário

9. regra :

   Se um relacionamento interage com quatro entidades
   Então subdividi-se o relacionamento em outros dois

10. regra :

    Se um relacionamento é do tipo " um para muitos "
    então uma instância de uma entidade interage com várias instâncias da outra

11. regra :

    Se um relacionamento é do tipo " um para um "
    então uma instância de uma entidade interage com uma instância da outra

12. regra :

    Se um relacionamento é do tipo " muitos para um "
    então várias instâncias de uma entidade interagem com uma instância da outra

13. regra :

    Se um relacionamento é do tipo " muitos para muitos "
    então várias instâncias de uma entidade interagem com várias instâncias da outra

14. regra :

    Se um relacionamento é do tipo "ternário"
    então o relacionamento é composto de três entidades

15. regra :

    Se três entidades possuem cardinalidade 1 com um relacionamento
    então o relacionamento é do tipo "ternário um para um"

Também foram geradas os seguintes informações:

- "comeriam" é um_para_muitos.
- "comeriam" é muitos_para_um.
- "comeriam" é um_para_um.
- "controla" é um_para_muitos.
- "controla" é muitos_para_um.
- "tem" é ternário_um_para_um.
- "tem-1" é um_para_um.
- "relacionam-se" é muitos_para_muitos.
- "cuida" é um_para_um.
- "paga" é um_para_um.

- "atende" é um_para_muitos.
- "atende" é muitos_para_um.
- "marcam " é muitos_para_muitos.
- "recebem" é muitos_para_muitos.
- "realizado" é um_para_um.
- "emite" é um_para_muitos.
- "emite" é muitos_para_um.
- "tem-1-1" é um_para_muitos.
- "tem-1-1" é muitos_para_um.
- "pertence" é um_para_um.
- "precisa" é um_para_um.
- "atenderia" é um_para_muitos.
- "atenderia" é muitos_para_um.

Não foram agregados mais atributos que os já existentes.

B.3 Modelo Textual

Antes da geração do modelo textual foi modificado o nome da entidade "setor de consultas" para "consultas" através da opção "Editar". O modelo textual gerado foi o seguinte:

- Entidade

  consulta "é do tipo" entidade.
  atendimento de médico "é do tipo" entidade.
  pagamento "é do tipo" entidade.
  médico "é do tipo" entidade.
  secretária "é do tipo" entidade.
  lanchonete "é do tipo" entidade.
  paciente "é do tipo" entidade.
  consultório "é do tipo" entidade.
  instituição "é do tipo" entidade.
  conta "é do tipo" entidade.
  carta de encaminhamento de paciente "é do tipo" entidade.
  relatório de caixa "é do tipo" entidade.

história de progresso de paciente "é do tipo" entidade.
registro de conta "é do tipo" entidade.
registro de dado "é do tipo" entidade.
cheque "é do tipo" entidade.

- Relacionamento

controla "é do tipo" relacionamento.
comeriam "é do tipo" relacionamento.
relacionam-se "é do tipo" relacionamento.
tem "é do tipo" relacionamento.
tem-1 "é do tipo" relacionamento.
tem-1-1 "é do tipo" relacionamento.
paga "é do tipo" relacionamento.
emite "é do tipo" relacionamento.
cuida "é do tipo" relacionamento.
atende "é do tipo" relacionamento.
recebem "é do tipo" relacionamento.
marcam "é do tipo" relacionamento.
precisa "é do tipo" relacionamento.
atenderia "é do tipo" relacionamento.
pertence "é do tipo" relacionamento.
realizado "é do tipo" relacionamento.

- Atributos

bonito "é do tipo" atributo.
histórico de paciente "é do tipo" atributo.
antecedente de família "é do tipo" atributo.
história de doença "é do tipo" atributo.
exame "é do tipo" atributo.
correto "é do tipo" atributo.
agradável "é do tipo" atributo.
rápido "é do tipo" atributo.
cadastro de doença "é do tipo" atributo.

- Relação entre atributos e entidades/relacionamento

consultório "possui o atributo" bonito.
consultório "possui o atributo" histórico de paciente.

secretária "possui o atributo" histórico de paciente.
consulta "possui o atributo" antecedente de família.
consulta "possui o atributo" história de doença.
consulta "possui o atributo" exame.
atendimento de médico "possui o atributo" correto.
atendimento de médico "possui o atributo" agradável.
atendimento de médico "possui o atributo" rápido.
secretária "possui o atributo" cadastro de doença.
consulta "possui o atributo" cadastro de doença.

- Relação

lanchonete "tem relação" 1 com precisa.
consultório "tem relação" 1 com precisa.
paciente "tem relação" 'N' com atenderia.
lanchonete "tem relação" 1 com atenderia.
secretária "tem relação" 1 com pertence.
relatório de caixa "tem relação" 1 com pertence.
história de progresso de paciente "tem relação" 'N' com tem-1-1.
registro de conta "tem relação" 1 com tem.
consulta "tem relação" 1 com tem-1-1.
consulta "tem relação" 1 com tem.
registro de dado "tem relação" 1 com tem.
consultório "tem relação" 1 com emite.
carta de encaminhamento de paciente "tem relação" 'N' com emite.
cheque "tem relação" 1 com realizado.
pagamento "tem relação" 1 com realizado.
atendimento de médico "tem relação" 'N' com recebem.
paciente "tem relação" 'N' com recebem.
consulta "tem relação" 'N' com marcam.
paciente "tem relação" 'N' com marcam.
conta "tem relação" 1 com paga.
secretária "tem relação" 1 com paga.
secretária "tem relação" 1 com cuida.
consultório "tem relação" 1 com cuida.
secretária "tem relação" 1 com atende.
paciente "tem relação" 'N' com atende.

paciente "tem relação" 'N' com relacionam-se.

instituição "tem relação" 'N' com relacionam-se.

secretária "tem relação" 1 com tem-1.

consultório "tem relação" 1 com tem-1.

secretária "tem relação" 1 com controla.

pagamento "tem relação" 'N' com controla.

médico "tem relação" 'N' com comeriam.

secretária "tem relação" 1 com comeriam.

lanchonete "tem relação" 1 com comeriam.

- Par_ordenado

secretária "através de" paga "relaciona-se com" conta "com tipos" 1 e 1.

consultório "através de" precisa "relaciona-se com" lanchonete "com tipos" 1 e 1.

lanchonete "através de" atenderia "relaciona-se com" paciente "com tipos" 1 e 'N'.

relatório de caixa "através de" pertence "relaciona-se com" secretária "com tipos" 1 e 1.

consulta "através de" tem "relaciona-se com" registro de conta "com tipos" 1 e 1.

consulta "através de" tem "relaciona-se com" registro de dado "com tipos" 1 e 1.

pagamento "através de" realizado "relaciona-se com" cheque "com tipos" 1 e 1.

paciente "através de" marcam "relaciona-se com" consulta "com tipos" 'N' e 'N'.

secretária "através de" cuida "relaciona-se com" consultório "com tipos" 1 e 1.

secretária "através de" atende "relaciona-se com" paciente "com tipos" 1 e 'N'.

instituição "através de" relacionam-se "relaciona-se com" paciente "com tipos" 'N' e 'N'.

consultório "através de" tem-1 "relaciona-se com" secretária "com tipos" 1 e 1.

secretária "através de" controla "relaciona-se com" pagamento "com tipos" 1 e 'N'.

secretária "através de" comeriam "relaciona-se com" lanchonete "com tipos" 1 e 1.

médico "através de" comeriam "relaciona-se com" lanchonete "com tipos" 'N' e 1.

paciente "através de" recebem "relaciona-se com" atendimento de médico "com tipos' 'N'e 'N'.

consulta "através de" tem-1-1 "relaciona-se com" história de progresso de paciente "com tipos" 1 e 'N'.

consultório "através de" emite "relaciona-se com" carta de encaminhamento de paciente "com tipos" 1 e 'N'.

Finalmente, na Figura B.7 é apresentado o diagrama completo da especificação correspondente ao modelo textual gerado.

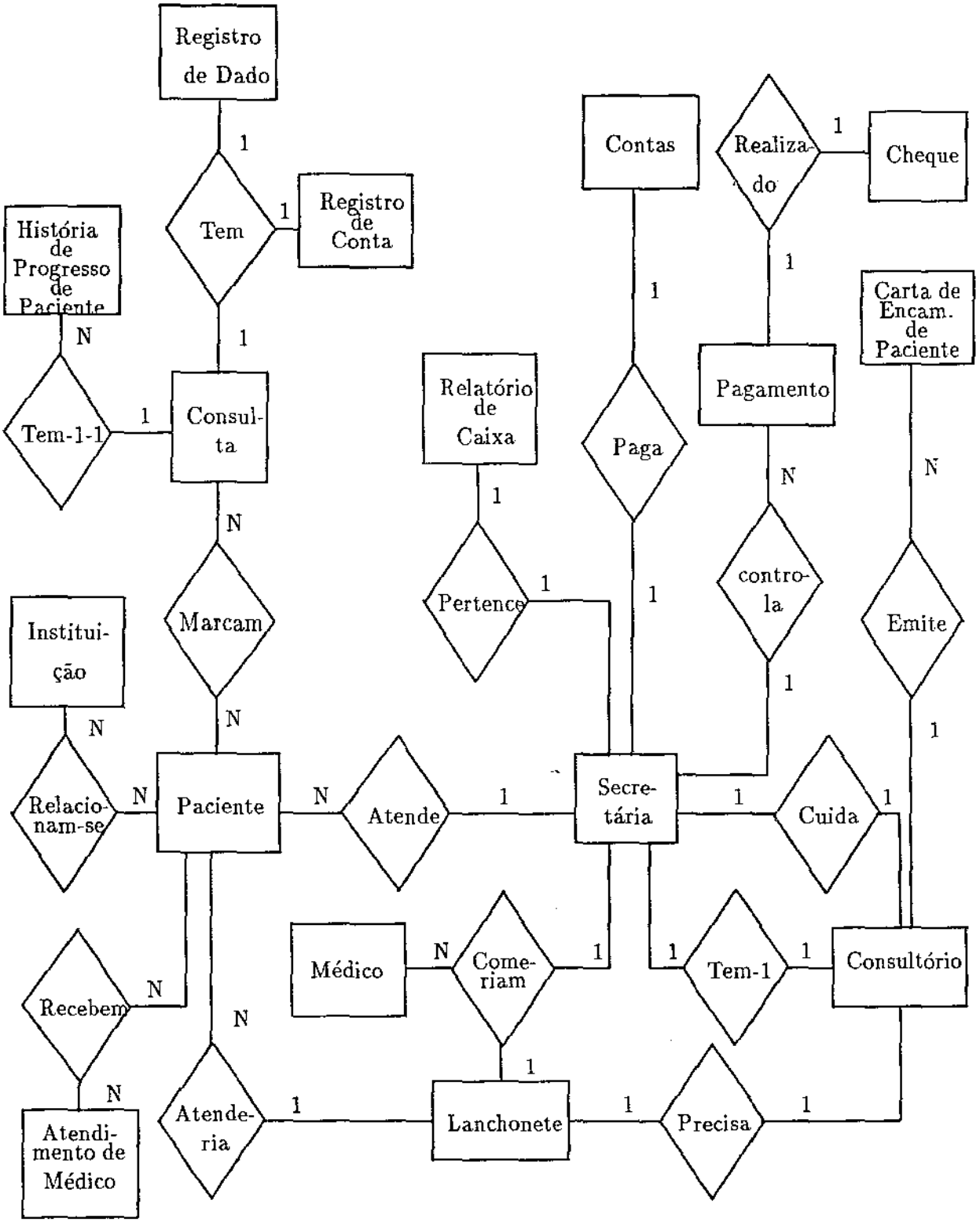


Figura B.7: Diagrama ER