Rodrigo Smarzaro da Silva

Ambiente Gráfico e Cooperativo para Gerência de Modelos

Dissertação apresentada ao Curso de Pós Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

Março de 2002

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Ambiente Gráfico e Cooperativo para a Gerência de Modelos

RODRIGO SMARZARO DA SILVA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. JOSÉ LUIS BRAGA – Orientador
Departamento de Informática – UFV

Prof. HENRIQUE PACCA LOUREIRO LUNA – Co-orientador
Departamento Ciência da Computação - ICEx - UFMG

Prof. HELENO DO NASCIMENTO SANTOS
Departamento de Informática – UFV

Prof. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação – ICEx – UFMG

Belo Horizonte, 22 de março de 2002.

Agradecimentos

Aos meus pais, Alcides Pereira da Silva e Maria Smarzaro da Silva por todo apoio e incentivo.

À minha mulher, Guanaeli Matias de Mendonça da Silva, pela paciência e colaboração.

À minha filha Júlia Mendonça Silva, pelos momentos de alegria em vários momentos de stress.

Aos meus amigos de Viçosa: Alessandro, pela companhia e solidariedade nas horas difíceis, e o casal Vítor e Andréia, pelas temporadas em que passei hospedado em sua casa.

Aos meus amigos em Linhares: Diogo, Paulo e Airton, por todos os momentos de esporte e descontração.

Ao meu amigo e orientador, José Luis Braga, por todo apoio e ajuda durante todo o mestrado.

À FAPEMIG, pela ajuda financeira.

Abstract

Despite the need for model use and management in organizations, a small number of decision makers use some tool for that purpose. Some reasons for this paradox are model complexity and the lack of a friendly environment for model construction, testing and management tailored for decision makers. In this work, we present GMMS (Graphical Model Management System), a graphical environment in which decision makers can formulate, manage and execute models in a simple and straight fashion. The environment provides support for SML (Structured Modeling Language), a model definition language, and for SQL-M, that is an SQL (Structured Query Language) extension for model management. Both languages are used through an easy to use graphical language. Models are created visually by dragging and dropping icons from a modeling palette, and model instances are created using an wizard that helps the decision maker along the process. Models are executed by using solvers external to the environment, in a way that is fully transparent to the decision maker. GMMS provides a friendly and easy to use environment where decision makers can play with models without having a deep technical knowledge on model management. This leaves spare time to get more concentrated on the problem and its domain.

Resumo

Apesar da necessidade de gerenciamento de modelos nas organizações, um número reduzido de tomadores de decisões utilizam alguma ferramenta para gerenciamento de modelos. Algumas razões para este paradoxo são a complexidade inerente aos modelos e a falta de um ambiente amigável para os tomadores de decisões. Neste trabalho, é proposto um ambiente gráfico onde os tomadores de decisão possam formular, gerenciar e executar modelos de forma simples e objetiva. O ambiente possui suporte à SML (*Structured Modeling Language*), como linguagem de definição de modelos, e à SQL-M, uma extensão da SQL (*Structured Query Language*) para gerenciamento de modelos. As duas linguagens são utilizadas de forma visual. Os modelos são criados, visualmente, através de grafos de modelos e as instâncias de modelos são criadas através de um assistente que auxilia o tomador de decisão em todo o processo. A execução do modelo é feita por resolvedores externos ao ambiente, mas tudo de forma transparente para o tomador de decisão. Todas estas características formam um ambiente, onde o tomador de decisão pode gerenciar os modelos necessários à organização de forma simples, sem a necessidade de conhecer profundamente uma linguagem de definição ou de manipulação de modelos.

Sumário

Lista de Figuras

Lista de Tabelas

1 – Introdução

Depara-se hoje com um volume de informações de complexidade muito maior do que a realidade de alguns anos. Esta complexidade deve-se a vários fatores como aumento do volume de informações disponível, cenários de disputa globais, aumento da capacidade computacional e disseminação de redes e internet, competição mais acirrada e mais complexa entre empresas, entre outros. Neste ambiente, onde uma decisão pode determinar o sucesso ou o fracasso de uma empresa, faz-se necessária a utilização de ferramentas de auxílio na tomada de decisão, que permitam aos tomadores de decisão gerenciar essa complexidade.

Apesar de as empresas estarem utilizando cada vez mais a tecnologia de informação e de comunicações para facilitar o acesso aos dados, os gerentes raramente conseguem aproveitar toda informação disponível e geralmente, utilizam-na apenas para tomar decisões simples. Com o intuito de alterar este fato, várias tecnologias têm sido desenvolvidas, tais como os Sistemas de Apoio a Decisão (SAD), Sistemas de Apoio a Decisão em Grupo (SADG), Sistemas de Informação Executiva (SIE), Sistemas Especialistas (SE), e Redes Neurais Artificiais (RNA) [Turban98].

Dentre estas, talvez a que tenha uso mais difundido dentro das empresas são os Sistemas de Apoio a Decisão. Os SAD são sistemas interativos, flexíveis e adaptáveis, baseados em computador, especialmente desenvolvidos para solucionar algum problema gerencial, com o objetivo de melhorar a tomada de decisão. Devem utilizar dados, possuir uma interface amigável, e são compostos basicamente por três subsistemas: Gerenciamento de Dados, Gerenciamento de Modelos e Comunicação (Interface). Um

quarto subsistema opcional que pode fazer parte de um SAD é o de Gerenciamento de Conhecimento, neste caso, o SAD será chamado de SAD inteligente, ou SAD baseado em conhecimento [Turban98].

O Gerenciamento de Dados e a Comunicação são tecnologias que já estão bastante consolidadas [Blanni93]. Atualmente, os responsáveis pelo Gerenciamento de Dados são os Sistemas de Gerenciamento de Banco de Dados (SGBD). Os SGBD atuais já permitem a fácil manipulação dos dados até mesmo por usuários não especialistas.

O Gerenciamento de Modelos surgiu da constatação de que os modelos também são informação estratégica e, sendo assim, deveriam ser gerenciados e receber atenção semelhante à dispensada aos dados e informações. Mas, ao contrário dos dados que possuem os SGBD para seu gerenciamento, o gerenciamento de modelos ainda não possui uma solução similar, universal. O grande problema é a complexidade que o gerenciamento de modelos envolve, geralmente necessitando de funcionários altamente especializados para que seja feito com eficiência [Blanni93], o que ocasiona o afastamento do principal utilizador dos modelos, que são os gerentes responsáveis pelas tomadas de decisões.

Para solucionar este problema, começaram a ser desenvolvidos Sistemas de Gerenciamento de Modelos (SGM), cujo principal objetivo é tornar a organização e a execução dos modelos fácil e transparente para os usuários, da mesma maneira que os Sistemas de Gerenciamento de Dados atuam em relação aos dados [Blanni93].

Um grande problema encontrado no desenvolvimento dos SGM é o formato de representação dos modelos. Até agora, a pesquisa mais promissora neste aspecto é a Modelagem Estruturada, introduzida por Arthur M. Geoffrion [Geoffr87], que fornece uma forte base teórica para a concepção, representação e manipulação de uma grande variedade de modelos. Um de seus principais objetivos é permitir que trabalhos baseados em modelos sejam feitos com ganho de produtividade e aceitação por usuários não especialistas. Por este motivo, a Modelagem Estruturada tem sido utilizada como base para a construção de protótipos de ferramentas para gerenciamento de modelos.

Outra dificuldade encontrada nos SGM é a obtenção dos dados para a instanciação dos modelos. Geralmente os ambientes dos SGM possuem uma linguagem própria para o gerenciamento de modelos, e devem efetuar chamadas externas para um SGBD para obtenção dos dados [Elmasri94]. Para solucionar este problema, Denílson A. Pereira [Pereir97] desenvolveu uma extensão da linguagem SQL, para uniformizar o tratamento entre modelos e dados, chamada SQL-M.

Com objetivo de resolver os problemas e superar as dificuldades citados, e de aproximar cada vez mais o usuário final, ou seja, o gerente, do ambiente de modelagem, a proposta deste trabalho é a criação de um ambiente gráfico para gerência de modelos que auxilie o gerente a formular, desenvolver e utilizar seus próprios modelos, ou modelos pré-existentes, utilizando a Modelagem Estruturada como forma de representação dos modelos, através de grafos de modelos (*Model Graphs*) e SML (*Structured Modeling Language*) e utilizando a SQL-M para a manipulação e instanciação dos modelos.

Neste ambiente, o usuário terá a seu dispor uma interface gráfica amigável para formulação e instanciação de modelos. Para fornecer uma maior segurança aos usuários, uma série de verificações de consistência são efetuadas. Utilizando os grafos de modelos, o usuário poderá formular visualmente suas classes de modelos, que depois podem ser instanciadas facilmente pelo uso da SQL-M.

Para a utilização da SQL-M, também são utilizados recursos gráficos semelhantes aos oferecidos pelo QBE (*Query By Example*), usado para facilitar o uso de SQL nos SGBD. Nesta versão de QBE, a interface do modelo é reconhecida e o usuário poderá então fornecer o conjunto de dados que servirá de entrada para criar a instância do modelo.

Para demonstrar a validade do trabalho proposto, foi implementado um protótipo, chamado GMMS (*Graphical Model Management System*), que permite a criação e instanciação de modelos de forma gráfica, facilitando sua utilização por usuários não especialistas.

O restante deste trabalho está organizado da seguinte maneira. O capítulo dois fará uma revisão da literatura sobre os SAD, com um enfoque maior em Gerência de Modelos, mostrando os principais sistemas e abordagens existentes atualmente. O capítulo três mostra uma visão geral do GMMS, incluindo um cenário de uso. O capítulo quatro mostra detalhes sobre a implementação do GMMS. Finalmente, o capítulo cinco mostra as conclusões e sugestões para a continuidade do trabalho.

2 – Revisão Bibliográfica

2.1 - Sistemas de Apoio ao Gerenciamento (SAG)

Segundo Turban [Turban98]:

> *"Os Sistemas de Apoio ao Gerenciamento referem-se a um conjunto de tecnologias computadorizadas cujo objetivo é apoiar o trabalho gerencial, em especial a tomada de decisão".*

Fazem parte deste conjunto de tecnologias os Sistemas de Informação Gerencial, os Sistemas de Apoio à Decisão, os Sistemas de Apoio à Decisão em Grupo, os Sistemas de Informação Executiva, os Sistemas Especialistas, e as Redes Neurais Artificiais. Todos estes sistemas foram criados com o objetivo de melhorar a eficácia da tomada de decisão gerencial, especialmente em casos complexos.

Os tipos de problemas que um SAG pode ajudar a resolver estão divididos em três categorias [Turban98]:

1. **Estruturados** – problemas repetitivos ou rotineiros que possuem uma solução padrão (ex. minimização de custos, maximização de lucros, etc).
2. **Semi-estruturados** – problemas complexos que possuem uma solução padrão para alguma de suas subpartes (ex. agendamento de produção, controle de estoque, avaliação de crédito, etc).
3. **Não-estruturados** – problemas complexos que não possuem uma solução padrão (ex. aprovação de empréstimos, aquisição de software, recrutamento de um executivo, etc).

2.1.1 – Sistemas de Informação Gerencial (SIG)

Segundo Keen e Morton [Keen78], os Sistemas de Informação Gerencial são designados para as tarefas estruturadas e repetitivas, onde procedimentos operacionais padronizados, regras de decisão e um fluxo de informações podem ser pré-definidos. Sua principal vantagem está na melhoria da eficiência, reduzindo custos, tempo e funcionários. A sua contribuição para a tomada de decisão é indireta, através do fornecimento de relatórios e acesso aos dados. São relativamente inflexíveis, sendo necessária reprogramação para extração de dados ou informações não previstas no projeto inicial. Seus usuários normalmente são gerentes até o nível médio.

2.1.2 – Sistemas de Apoio à Decisão (SAD)

Sistemas de Apoio à Decisão são sistemas baseados em computador que auxiliam os gerentes nos processos de tomada de decisão de problemas semi-estruturados e não-estruturados. Suas aplicações são basicamente de planejamento estratégico de longo prazo. Sua principal meta é auxiliar a eficácia da decisão de analistas e gerentes, que são seus principais usuários.

Um SAD pode fornecer representações válidas de sistemas do mundo real, especialmente problemas *ad-hoc*, isto é, problemas inesperados e não repetitivos dentro de um curto espaço de tempo. Ele pode evoluir à medida que o gerente aprende mais sobre o problema e pode ser desenvolvido por profissionais fora da área de processamento de dados. Uma característica marcante dos SAD é o seu alto nível de flexibilidade, permitindo extração de informações com facilidade, normalmente sem necessidade de reprogramação, via interfaces de uso fácil e intuitivo para equipes técnicas.

2.1.3 – Sistemas de Apoio à Decisão em Grupo (SADG)

Segundo Huber [Huber84], um SADG consiste de um conjunto de software, hardware, componentes de linguagem, e procedimentos que apóiam um grupo de pessoas engajadas em uma reunião para a tomada de decisões.

Segundo Gallupe e DeSanctis [Gallup88], SADG é um sistema interativo baseado em computador que facilita a solução de problemas não-estruturados por um grupo de tomadores de decisão.

Turban [Turban98] descreve um SADG como um SAD distribuído, possibilitando, assim, que vários grupos, possivelmente espalhados geograficamente, tomem decisões em conjunto, partilhando os mesmos dados, condições de contorno e modelos. Um dos requisitos de um SADG é a coordenação de todo o trabalho de decisão, controle de versões, travamento para atualização, controle de acesso e vários outros requisitos de sistemas CSCW (*Computer Supported Cooperative Work*).

O objetivo de um SADG, segundo Finholt e Sproull [Finholt90], é melhorar a produtividade e eficácia das reuniões de tomada de decisão, através da aceleração do processo decisório ou melhorando a qualidade das decisões resultantes. Isto é feito fornecendo suporte à troca de idéias, opiniões e preferências dentro do grupo.

2.1.4 – Sistemas de Informação Executiva (SIE)

Os Sistemas de Informação Executiva são sistemas baseados em computador que suprem as necessidades de informação de executivos dos níveis hierárquicos mais altos. Fornecem rápido acesso a informações e dados tanto internos quanto do ambiente do sistema, e a relatórios gerenciais. Possuem interface bastante amigável, utilizando gráficos, fornecem relatórios de exceções e possuem capacidade de quebrar os dados em detalhes mais específicos, como, por exemplo, decompor um relatório de vendas em vendas por região, por produto, ou por vendedor. Esta capacidade de quebrar os dados em detalhes recebe o nome de *"drill-down"* e permite ao usuário do sistema identificar problemas e oportunidades de negócio.

2.1.5 – Sistemas Especialistas (SE)

Um Sistema Especialista é, tipicamente, um sistema decisório ou de resolução de problemas que pode alcançar um nível de performance comparável, ou mesmo excedendo, o nível de um especialista humano em alguma área especializada e específica [Turban98].

A idéia básica de funcionamento de um Sistema Especialista é simples, consistindo na transferência da perícia de um especialista humano para o computador. Este conhecimento armazenado pode então ser consultado sempre que necessário por usuários não especialistas. O computador realiza inferências até chegar a uma conclusão específica. Depois, assim como um consultor humano, sugere possíveis soluções ou

diagnósticos ao usuário não especialista e se necessário, explica a lógica por trás de cada sugestão.

2.1.6 – Redes Neurais Artificiais (RNA)

Ao contrário dos sistemas anteriores, que estavam baseados no uso de informações, dados ou conhecimento explícitos armazenados em um computador, as Redes Neurais Artificiais estão baseadas em tecnologia de reconhecimento de padrões, que capacitam o computador com mecanismos de aprendizado para tratar dos problemas do mundo real que são baseados em informações parciais, incompletas, ou inexatas [Turban98].

As RNA são uma tecnologia recente, e muita pesquisa ainda deve ser feita nesta área, embora alguns resultados expressivos já possam ser vistos em aplicações comerciais. Uma vez treinadas para reconhecer um determinado tipo de padrão desejado, as RNA conseguem determinar se um outro conjunto de dados fornecido como entradas é aderente ao padrão aprendido, ou não. As aplicações comerciais desse tipo de ferramenta ainda são incipientes, mas podem ser citadas: segurança, detecção de fraudes, classificação de clientes em categorias, e outras.

2.1.7 – Sistemas de Apoio Híbridos

Todos os SAG citados, embora tenham seus usos específicos, podem ser combinados e formar o que é chamado de um Sistema de Apoio Híbrido. Este apoio de um sistema ao outro pode ser feito de várias maneiras, como por exemplo, um SE ser usado para melhorar a modelagem e a gerência de dados de um SAD, uma RNA ou um SADG ser usado para apoiar a aquisição de conhecimento para a construção de um SE. Uma forte tendência que tem sido observada é a utilização de SE apoiando os outros sistemas, na tentativa de torná-los mais "espertos".

2.2 – Sistemas de Apoio à Decisão

O movimento a favor dos Sistemas de Apoio à Decisão começou a partir da identificação de deficiências em outras áreas de tecnologia de gerenciamento como Pesquisa Operacional, ciência de gerenciamento (OR/MS) e sistemas de gerenciamento de informações. As duas principais deficiências eram não conseguir satisfazer a crescente demanda dos gerentes de um suporte mais efetivo para a tomada de decisão e

não conseguir um uso apropriado da tecnologia de processamento de informação, especialmente em áreas como gerência de informação e computação interativa [Blanni93].

O principal conceito por trás da Pesquisa Operacional e da ciência de gerenciamento é a construção de modelos de decisão e o desenvolvimento de técnicas de solução para estes modelos (por exemplo, programação matemática e processos estocásticos). O problema é que não foi dada muita atenção ao modo como estes modelos eram implementados e, praticamente, não foi pensado em como os gerentes iriam utilizar estes modelos futuramente [Blanni93].

O primeiro trabalho que tratou esta deficiência foi de Michael Scott Morton [Morton71] em 1971 sob o título de Sistemas de Gerenciamento de Decisão. Morton propôs um modelo de produção/distribuição onde os gerentes poderiam fazer análises *what-if* em possíveis mudanças na produção, distribuição e marketing. Ao implementar este modelo, Morton cuidou de dois problemas que não estavam sendo tratados anteriormente. O primeiro era o uso de uma interface mais amigável (utilizando uma abordagem gráfica) entre o gerente e o SAD. O segundo foi o impacto do sistema na eficácia das decisões tomadas pelos gerentes [Blanni93].

O principal conceito na área de sistemas de gerenciamento de informação é de que ela trata de processos de decisão bem estruturados, ao invés de processos semi-estruturados ou não-estruturados. Devido a esta deficiência, foi proposto por Gorry e Morton [Gorry71], em 1971, melhorar a tecnologia de sistemas de gerenciamento de informações, incluindo os processos semi-estruturados e não-estruturados no seu escopo de atuação [Blanni93].

Somente sete anos mais tarde, Keen e Morton [Keen78] publicaram um livro sobre SAD. Eles identificaram três objetivos principais para esse tipo de sistema:

1 – ajudar gerentes na tomada de decisão de processos semi-estruturados;

2 – ajudar, ao invés de substituir o julgamento do gerente;

3 – melhorar a eficácia da decisão ao invés da eficiência

Um dos fatos que impulsionaram as pesquisas iniciais na área de SAD, foi o progresso na gerência de dados no final da década de 60, e seu posterior estabelecimento como uma área produtiva de pesquisa e prática no final da década de 70. Vários trabalhos, que surgiram durante este período, tratavam de estender a capacidade das ferramentas de gerência de dados para que suportassem modelos de decisão e uma interface mais amigável com o usuário [Blanni93]. Bonczek, Holsapple e

Whinston [Boncze82] descrevem um SAD como uma evolução da gerência de dados para a gerência de modelos, e sugerem que um SAD consiste de um sistema de conhecimento (dados, modelos, relacionamentos), um sistema de processamento (procedimentos para organizar, manter e processar os tipos de informação) e um sistema de linguagem para fazer o papel de interface com o usuário.

Para Sprague e Carlson [Spragu82], a estrutura de um SAD é composta por três funções prestadas ao usuário: Gerência da Base de Dados, Gerência de Modelos e Gerência de Diálogos.

Turban [Turban98] enumera uma quarta função opcional ao SAD chamada Gerência de Conhecimento.

Um modelo conceitual de um SAD pode ser visto na Figura 2.1. Uma descrição mais detalhada sobre cada uma das quatro funções (subsistemas) de um SAD segue abaixo:

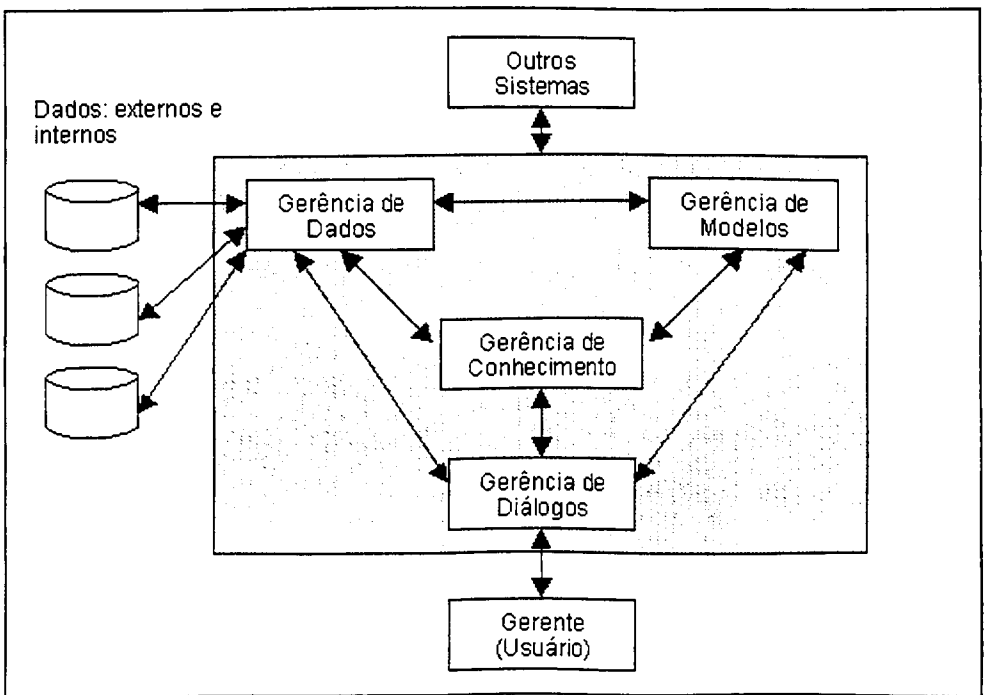


Figura 2.1 – Modelo Conceitual de um SAD (adaptado de [Turban98])

- **Gerência da Base de Dados** – inclui o banco de dados, que contém dados relevantes ao SAD e é gerenciado por um SGBD, que tem funções de capturar e extrair dados para inclusão no banco de dados do SAD, atualizações rápidas, inter-relacionar dados de diferentes fontes, recuperação rápida dos dados para consultas e relatórios, proteção de uso

não autorizado dos dados, realizar recuperação e manipulação complexa de dados baseadas em consultas e manter histórico de uso dos dados.

- **Gerência de Modelos** – composto por uma base de modelos, uma linguagem de modelagem, um diretório de modelos, comandos para integração e execução dos modelos, e um sistema de gerenciamento de modelos que deve ser capaz de: criar modelos de modo fácil e rápido, permitir aos usuários manipular, armazenar e gerenciar diferentes tipos de modelos de maneira lógica e integrada, catalogar e mostrar o diretório de modelos para uso de vários indivíduos em uma organização, inter-relacionar modelos com o banco de dados, gerenciar e manter os modelos com funções semelhantes à gerência de dados.
- **Gerência de Diálogos** – é todo software ou hardware que fornece algum tipo de interface do usuário com o SAD. Inclui fatores como facilidade de uso, acessibilidade, e interação homem-máquina. Alguns especialistas em SAD afirmam que a interface é o componente mais importante, devido ao fato de que grande parte do poder, flexibilidade, e facilidade de uso de um SAD estarem intimamente ligados a este componente [Blanni93]. Uma interface não apropriada é uma das grandes causas para a resistência dos gerentes em usar computadores e análises quantitativas.
- **Gerência de Conhecimento** – Alguns problemas não-estruturados e semi-estruturados são tão complexos que exigem perícia para a sua solução. Tal perícia pode ser fornecida através de um SE. Portanto, os SAD mais avançados estão equipados com um subsistema de gerência de conhecimento que é o responsável por fornecer a perícia necessária para resolver alguns aspectos de um problema, ou fornecer conhecimento para melhorar a performance de outros componentes do SAD. A gerência de conhecimento é composta de um ou mais SE que, assim como a gerência de dados e modelos, fornece mecanismos necessários para a execução e integração dos SE. Um SAD que possui um componente de gerência de conhecimento também é conhecido como SAD inteligente, SAD/SE, ou SAD baseado em conhecimento.

Este trabalho está inserido diretamente no contexto da Gerência de Modelos que será melhor detalhada na seção seguinte.

2.3 – Gerência de Modelos

Para um melhor entendimento sobre os Sistemas de Gerenciamento de Modelos (SGM), será feita uma caracterização dos modelos e dos diferentes paradigmas de modelagem. Em seguida será apresentado um breve histórico de evolução dos SGM e, logo após, serão mostradas as características que um SGM deve possuir.

2.3.1 – Modelos

Nas pesquisas na área de SAD, um de seus componentes que recebe maior atenção é a gerência de modelos. Um motivo para esta atenção é o fato da existência da gerência de modelos estar intimamente ligada com a existência dos SAD. Ao contrário dos outros componentes, gerência de dados e gerência de diálogos, que possuem muitas aplicações fora do ambiente dos SAD, a gerência de modelos foi criada devido à necessidade de suporte aos processos de decisão [Blanni93].

Dentro da gerência de modelos, o componente fundamental é o próprio modelo, por isso, faz-se necessária uma definição para o termo "modelo". A área de gerência de modelos é muito nova para uma terminologia consistente e em razão disto, vários pesquisadores parecem ter diferentes concepções sobre o termo "modelo" [Tsai01]. Para eliminar o problema de diferentes conceitos, será adotado neste trabalho o mesmo conceito utilizado por Muhanna [Muhann94], isto é, uma representação abstrata de algum problema do mundo real. Para o propósito deste trabalho, esta definição receberá um complemento, e ficará como segue:

- **Modelo** – Uma representação abstrata de algum problema do mundo real [Muhann94] que pode ser acoplada à um dispositivo que transforma dados em informação útil para um tomador de decisão [Turban98].

Da mesma maneira, outros termos relacionados a modelos também terão seu significado estabelecido como segue:

- **Esquema de modelo** – utilizado para designar uma representação formal ou semiformal de alguma classe de problemas do mundo real.
- **Instância de modelo** – especifica uma instância ou caso particular de uma classe de problemas. Uma instância de modelo é formada atribuindo dados específicos às variáveis e/ou coeficientes para um esquema de modelo.

- **Resolvedor** – consiste em um procedimento que deve ser executado (por computador ou não) para resolver uma determinada instância de modelo e determinar as soluções procuradas.

Como exemplo, considere o problema de encontrar a produção ótima em um período para uma empresa produzindo m tipos de produto sujeito a restrições de recursos dados por n fatores de produção, e que a produção não deve exceder a demanda. Esse problema pode ser resolvido utilizando-se um modelo de Programação Linear. Neste caso, pode-se especificar o seguinte esquema de modelo para o problema real:

$$\begin{array}{ll} \text{Maximize} & z = cx \\ \text{Sujeito a:} & Ax \leq b \\ & Ix \leq s \\ & x \geq 0 \end{array} \qquad \text{(exemplo 1)}$$

onde:

c = vetor de dimensão m de margem de lucro dos produtos.

x = vetor de dimensão m de produção ótima.

b = vetor de dimensão n de recursos disponíveis.

Uma instância para esse modelo com base em um determinado conjunto de dados poderia ser a seguinte:

$$\begin{array}{ll} \text{Maximize} & z = 8x1 + 5x2 + 3x3 \\ \text{Sujeito a:} & x1 + 2x2 + 5x3 \leq 41 \\ & 3x1 + 4x2 + x3 \leq 25 \\ & 6x1 + 3x3 \leq 93 \\ & x1 \leq 10 \qquad x1 \geq 0 \\ & x2 \leq 5 \qquad x2 \geq 0 \\ & x3 \leq 9 \qquad x3 \geq 0 \end{array}$$

Para resolver esse modelo instanciado, pode-se usar um resolvedor baseado em alguma variante do algoritmo simplex como, por exemplo, o LINDO [Schrag99].

2.3.2 – Formas de Representação de Modelos

Existem diferentes tipos de modelos (otimização, simulação, descritivos, e outros), e diversas formas de representá-los (linguagens algébricas, planilhas eletrônicas, e outras). Devido à esta variedade de opções, uma das grandes dificuldades

no desenvolvimento de um SGM é a forma de representação dos modelos. Uma tendência de pensamento é imaginar que da mesma maneira que existem as DDL (*Data Definition Language*, ou *Data Description Language*) nos SGBD, também deve existir uma linguagem para descrição de modelos para os SGM [Tsai01].

Anterior aos trabalhos na área de gerência de modelos, os desenvolvedores de SAD criavam seus próprios modelos a partir do zero e desenvolviam geradores de matrizes para comunicação com estes modelos. Estes geradores de matrizes eram escritos para uma aplicação específica, e não eram adaptáveis facilmente para uma outra aplicação semelhante [Blanni93]. Além disso, seu uso necessitava de um profissional com experiência em modelagem, o que limitou seu uso entre os gerentes, que seriam os maiores usuários. Para contornar este problema, foram criadas linguagens de modelagem (ou linguagens de definição de modelos) e sistemas que utilizam uma notação algébrica, tais como SML [Geoffr90], UIMP [Elliso82], AMPL [Fourer90], LPL [Hürlim88], gLPS [Collau94], AIMMS [Bissch94], GAMS [Brooke92], MPL [Maxima93], LINGO [Schrag99b], e LINDO [Schrag99]. Atualmente, o uso de linguagens de modelagem para construir modelos está se expandindo, bem como a pesquisa neste campo. Através do seu uso, a maior parte do trabalho fica a cargo do computador, que deve ser capaz de reconhecer automaticamente a estrutura do modelo, dispensando o trabalho de tradução.

Independente de que linguagem é utilizada para especificar um modelo, alguns aspectos importantes devem ser levados em conta na hora da escolha de uma linguagem de modelagem. Tsai [Tsai01] descreve vários aspectos desejáveis em uma boa linguagem de definição de modelos, tais como:

- Ser capaz de representar uma estrutura de uma classe de modelos separadamente de qualquer instância do modelo.
- Ser não-procedimental para facilitar sua utilização.
- Possuir estrutura rigorosa e independente de notação para modelagem, o que força uma abordagem sistemática e cuidadosa para a representação do modelo.

2.3.3 – Sistemas Gerenciadores de Modelos (SGM)

As pesquisas na área de gerência de modelos começaram em 1975 sob a sugestão de que os modelos são um recurso organizacional importante, assim como os

dados comuns, e como tal devem ser desenvolvidas ferramentas para o seu gerenciamento [Spragu75], chamadas de Sistemas de Gerenciamento de Modelos (SGM) [Blanni93] [Chari98].

Os primeiros trabalhos na área de gerenciamento de modelos surgiram a partir de gerenciadores de bancos de dados baseados no modelo Codasyl DBTG. Alguns dos arquivos no sistema de gerenciamento de dados foram substituídos por modelos e a ligação, entre os arquivos e os modelos, ficou por conta do sistema de gerenciamento de modelos [Yeo96].

Apesar do reconhecimento da importância da modelagem nas organizações, ainda há muita ambigüidade em sua utilização. Os tomadores de decisão sabem da necessidade do uso de tecnologia de modelagem para as decisões, mas relutam em utilizá-la, pois o modelo resultante geralmente é incompreensível e intimidador [Dolk88].

O propósito de um Sistema de Gerenciamento de Modelos é tornar a organização e a execução dos modelos fácil e transparente para os usuários, da mesma maneira que os Sistemas de Gerenciamento de Dados atuam em relação aos dados [Blanni93]. Alguns trabalhos na gerência de modelos como os ambientes SYMMS [Muhann94] e GBMS/SM [Chari98] descrevem protótipos de gerenciadores de modelos. Uma deficiência nos dois ambientes citados é a necessidade de muito conhecimento e muita habilidade no uso de linguagens de modelagem e de consulta.

Uma abordagem para a representação de modelos é a utilização de conceitos de uma álgebra relacional, onde os modelos são vistos como um sistema de relações virtuais que não são armazenáveis. Os registros destas relações são gerados sob demanda, da mesma forma que uma consulta nos SGBD atuais. A integração de modelos é então feita pelo uso de operações de junção, projeção e seleção nas relações virtuais [Blanni85]. Um objetivo importante desta abordagem é conseguir chegar a uma estrutura única de trabalho para integrar a gerência de dados e a gerência de modelos [Blanni86].

Outra abordagem que merece destaque é a baseada em conhecimento. Nesta abordagem são usadas ferramentas e técnicas de inteligência artificial para a gerência de modelos. Uma variedade de esquemas de representação de conhecimento tais como redes semânticas, cálculo de predicados de primeira ordem e regras de produção têm sido utilizadas para a representação e gerenciamento dos modelos [Suh95] [Blanni93].

Uma das maiores inovações na área de modelagem foi proposta por Arthur M. Geoffrion [Geoffr87]. Ele desenvolveu o conceito de Modelagem Estruturada em que entidades do mundo real e suas relações são organizadas em um grafo acíclico direcionado, sendo em seguida mapeados em uma decomposição hierárquica de seus componentes.

Uma outra proposta interessante foi a extensão da linguagem SQL para a linguagem que recebeu o nome de SQL-M. Apesar da SQL-M não ser uma linguagem de modelagem, ela permite uma padronização no formato de como os modelos e os resolvedores são selecionados. Ainda permite a instanciação de modelos com independência dos dados e do resolvedor, além de fornecer mecanismos para composição de modelos. [Pereir97].

Apesar das diferentes abordagens para representação dos modelos em um SGM, ainda não existe uma padronização aceita. Entretanto, a pesquisa na área de Modelagem Estruturada tem ganhado força, e já é possível encontrar diversos protótipos que a utilizam (veja Seção 2.6).

Como a Modelagem Estruturada e a SQL-M são assuntos que fazem parte deste trabalho, cada uma delas será detalhada nas seções 2.4 e 2.5 respectivamente.

2.4 – Modelagem Estruturada

A Modelagem Estruturada foi desenvolvida para facilitar e integrar tantos aspectos do processo de modelagem quanto possível.

Segundo Geoffrion [Geoffr87]:

> *"A Modelagem Estruturada se esforça para fornecer um ambiente matemático formal, uma linguagem, e um ambiente baseado em computador para a criação, representação, e manipulação de uma grande variedade de modelos".*

A modelagem estruturada é uma estrutura unificada de modelagem baseada em grafos acíclicos para representar as referências cruzadas entre os elementos de um modelo e hierarquias para representar diferentes níveis de abstração. Fornece um formalismo rigoroso para a representação e oferece um terreno fértil para a implementação de Sistemas de Gerenciamento de Modelos.

Embora a motivação para o surgimento da Modelagem Estruturada tenha surgido de fatores como a baixa aceitação e produtividade de modelos MS/OR, da

melhoria da tecnologia de bancos de dados, e no aumento da popularidade de computadores pessoais e modelagem com planilhas [Geoffr87], muita fertilização cruzada ocorreu com outras disciplinas orientadas à modelos, como sistemas de bancos de dados, inteligência artificial, e engenharia de software. Como resultado, a Modelagem Estruturada evoluiu para uma abordagem muito genérica com potencial de aplicação em diversos campos. Essa generalidade a qualifica como uma ferramenta poderosa e integrada para a gerência de recursos de informação [Dolk88].

2.4.1 – Objetivos da Modelagem Estruturada

Os sistemas de modelagem do passado acabaram levantando mais barreiras do que removendo [Blanni93]. Um sistema de modelagem bem sucedido deve ser capaz de integrar tecnologias como bancos de dados e planilhas, para expandir o escopo e impacto das atividades de modelagem dentro de uma organização, o que não será possível a menos que se obtenha um ambiente que atenda aos seguintes requisitos [Geoffr87]:

- apresente estrutura conceitual que define uma estrutura geral de modelos;
- permita independência de representação de modelos para com os operadores de solução de modelos e com os dados associados com instâncias específicas de modelos;
- tenha capacidade de capturar uma grande variedade de modelos matemáticos, bem como outros modelos conceituais relacionados a disciplinas de projeto de base de dados e engenharia de software;
- ofereça suporte para o ciclo de vida de modelagem como um todo;
- utilize as facilidades de gerenciamento de modelos da mesma maneira que em sistemas de gerenciamento de dados.

A Modelagem Estruturada é uma abordagem genérica para problemas e atividades associadas com modelagem [Dolk88]. Os sistemas de modelagem anteriores a Modelagem Estruturada eram específicos para alguma aplicação, e não eram capazes de atender aos requisitos mencionados. Embora os objetivos sejam muito ambiciosos, eles são necessários para o sucesso de implementação de um Sistema de Gerenciamento de Modelos [Geoffr87].

2.4.2 – Fundamentos da Modelagem Estruturada

A Modelagem Estruturada é um ambiente unificado de modelagem, baseado em grafos acíclicos e direcionados para representar referências cruzadas entre elementos de um modelo, e hierarquias para representar diferentes níveis de abstração. Existem três estruturas básicas que se encaixam nesta definição [Geoffr87]:

- **Estrutura elementar** – Nível mais baixo que captura detalhes específicos de uma instância de um modelo [Geoffr92a], constituído por uma coleção de elementos acíclica, finita, fechada e não vazia.
- **Estrutura genérica** – captura as dependências de definição entre os gêneros, enquanto omite os detalhes de uma instância de modelo, ou seja, particiona a estrutura elementar de tal modo que exista apenas uma partição (*genus*) para cada tipo de elemento. O *genus* é semelhante à noção de conjunto, ou classe. O particionamento deve satisfazer a similaridade genérica, ou seja, cada elemento de um *genus* deve ter o mesmo número de segmentos na seqüência de chamada, e todos os elementos de um dado segmento de seqüência de chamada deve pertencer ao mesmo *genus*. O particionamento assegura uma tipagem forte de modo que cada elemento deve pertencer a um, e somente um, *genus*.
- **Estrutura modular** – uma árvore definida a partir da estrutura genérica, onde todas as folhas são *genera*, e todos os nodos não terminais são módulos. A estrutura modular permite que os *genera* possam ser agrupados de maneira que possam ter um significado conceitual para os usuários. Permite que o usuário veja o modelo de diferentes níveis de abstração. Uma estrutura modular deve satisfazer a ordenação monótona, isto é, aquelas que admitem uma representação em forma de lista indentada, com nenhuma referência encaminhada (*genera* que chama outro *genera* mais abaixo na lista).

Um modelo estruturado consiste em uma estrutura elementar, uma estrutura genérica que satisfaça a similaridade genérica, e uma estrutura modular com ordenação monótona.

Os modelos são representados em termos de elementos que podem ser particionados em gêneros e depois agregados em módulos. Existem 5 tipos de elementos:

- **Entidade Primitiva (pe)** – uma definição primitiva representando qualquer entidade distintamente identificável.
- **Entidade Composta (ce)** – uma definição baseada em outras definições provenientes de entidades primitivas ou de outras entidades compostas. Não necessitam de valor.
- **Atributo (a)** – uma definição que possui um valor fornecido pelo usuário em um certo intervalo, baseado na definição de alguma entidade primitiva ou composta. Possui uma variação chamada de **Atributo Variável (va)** que é um atributo cujo valor o modelador espera mudar freqüentemente, ou colocar sob controle do resolvedor.
- **Função (f)** – uma definição com um certo valor calculado por uma certa regra, baseada nas definições de certos outros elementos.
- **Teste (t)** – Uma definição com um valor lógico calculado por uma certa regra, baseada nas definições de certos outros elementos.

Cada elemento possui uma seqüência de chamada, que é uma tupla contendo todos os elementos de que um elemento não primitivo depende. A seqüência de chamada captura as referências cruzadas entre elementos do modelo, e pode ser derivada diretamente da representação gráfica do modelo.

2.4.3 – Exemplo: Modelo de Transporte

Para exemplificar a representação de um modelo utilizando a Modelagem Estruturada, utilizaremos um modelo, bastante conhecido, chamado Modelo de Transporte (*Transportation Model*). Dois fatores influenciaram a escolha deste modelo para o exemplo. Primeiro, por se tratar de um modelo bastante conhecido e presente em quase todos os textos introdutórios de OR/MS. Segundo, por sua estrutura representada em modelagem estruturada apresentar os cinco tipos (pe, ce, a, f, t), além da variação do elemento atributo (va). O cenário deste Modelo de Transporte é composto de fábricas (*Plants*) que produzem um único produto para remessa (*Shipment*) aos clientes (*Customers*). Cada fábrica possui uma capacidade máxima de fornecimento (*Maximum Supply Capacity*), e cada cliente tem uma necessidade de demanda (*Demand*) exata.

Para cada ligação (*LINK*) que existe entre uma fábrica e um cliente, existe um custo unitário de transporte associado (*Unity Flow Cost*). O modelo permite que sejam avaliados vários fluxos de transporte, em relação ao custo total de transporte (*Total Cost*), entre as ligações que satisfazem as capacidades de produção, e os requisitos de demanda.

Neste exemplo, as entidades primitivas incluirão cada instância das fábricas (ex. fábricas em Belo Horizonte e Vitória), e cada instância dos consumidores (ex. consumidores em São Paulo, Rio de Janeiro, e Linhares). Os atributos compostos incluíram as ligações entre as fábricas e os consumidores (ex. BH-SP, BH-RJ, BH-Linhares, Vitória-SP, Vitória-RJ, Vitória-Linhares). Os atributos incluem a capacidade de fornecimento de cada fábrica, os requisitos de demanda para cada consumidor, e o custo de transporte de cada ligação. O fluxo de cada ligação pode ser um atributo variável. Os elementos de teste definem as restrições de fornecimento de cada fábrica, e as restrições de demanda de cada consumidor. Um único elemento de função calcula o custo total de transporte. A Figura 2.3 apresenta a Estrutura Elementar do modelo de transporte.

Como pode ser observada na Figura 2.3, a estrutura elementar captura todas as associações entre os elementos anteriormente descritos. Para facilitar a visualização do modelo, pode-se particionar a estrutura elementar em uma estrutura genérica, como foi dito anteriormente na seção 2.4.2, definindo os seguintes *genera*:

- **Entidades Primitiva** – Fábrica (*PLANT*) e Consumidor (*CUSTOMER*).
- **Entidade Composta** – ligações (*LINK*)
- **Atributos** – Fornecimento (*SUPPLY*), Demanda (*DEMAND*), Custo (*COST*).
- **Atributo Variável** – Fluxo (*FLOW*).
- **Testes** – Teste de fornecimento (*T:SUP*) e teste de demanda (*T:DEM*).
- **Função** – Custo Total (*TOTALCOST*).

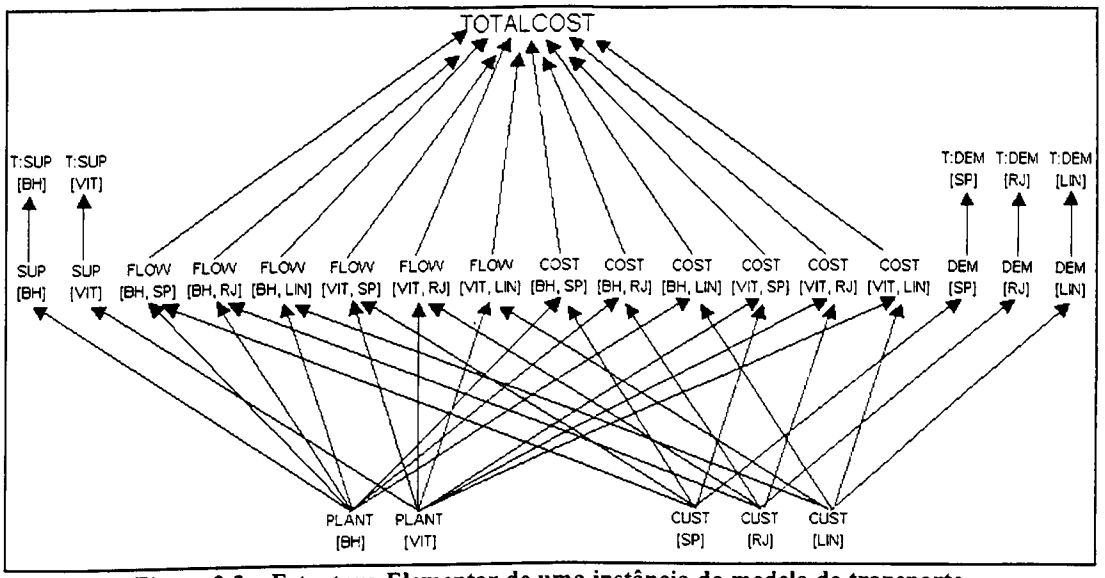


Figura 2.3 – Estrutura Elementar de uma instância do modelo de transporte

A Figura 2.4 mostra a estrutura genérica do modelo de transporte, derivada da estrutura elementar mostrada na Figura 2.3. A similaridade assegura que as seqüências de chamada para os *genera* devem ser iguais às dos elementos. Nota-se que a representação gráfica para a estrutura genérica é muito mais concisa e significativa que a estrutura elementar que, por ser muito detalhada, raramente é de muito uso.

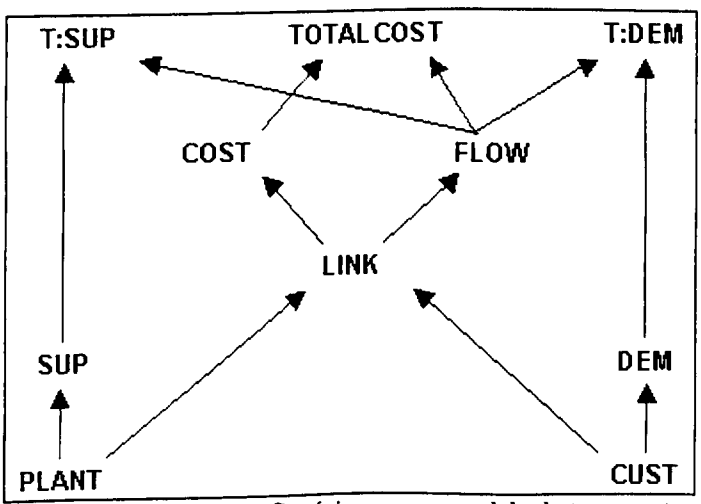


Figura 2.4 – Estrutura Genérica para o modelo de transporte

Além do particionamento realizado, pode-se ainda agrupar os *genera* relacionados com o intuito de representar diferentes níveis de abstração. Por exemplo, no modelo de transporte, os *genera* podem ser agrupados em componentes de vendas (&SALES), produção (&PROD) e distribuição (&DIST). O símbolo "&" antes dos nomes define um módulo na modelagem estruturada, que pode ser visto como uma árvore cujas folhas são *genera*. A Figura 2.5a mostra a estrutura modular do modelo de

transporte. Nota-se que a ordenação monótona da estrutura modular preserva a aciclicidade das estruturas genéricas correspondentes. A estrutura modular pode ser representada ainda como um contorno modular, onde um caminhamento pré-ordem é efetuado na estrutura modular, e os diferentes níveis de indentação correspondem aos diferentes níveis de hierarquia da estrutura modular. A Figura 2.5b mostra o contorno modular para o modelo de transporte.

Um esquema de modelo pode ser formado a partir do contorno modular, expandindo-se cada linha em um parágrafo de acordo com a seguinte sintaxe [Geoffr90]:

(Estruturas entre colchetes são opcionais)

Para elementos do tipo entidade primitiva:

GNAME [new index] /pe/ [index set statement] [domain statement] [interpretation]

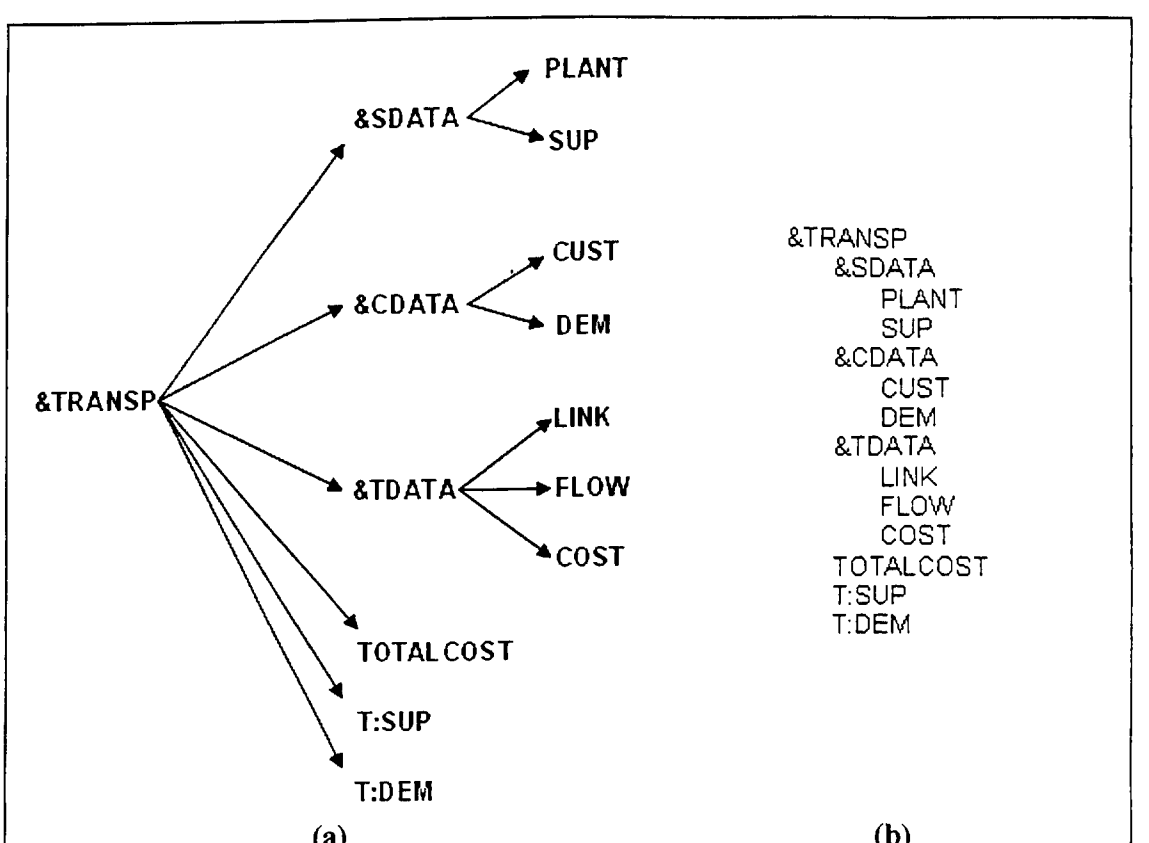


Figura 2.5 – (a) Estrutura Modular, e (b) Contorno Modular do modelo de transporte.

Para elementos do tipo entidade composta:

GNAME[new index] (generic calling sequence) /ce/ [index set statement] [domain statement] [interpretation]

Para elementos do tipo atributo:

GNAME[new index] (generic calling sequence) /a/ [index set statement] [domain statement] [range statement] [interpretation]

Para elementos do tipo atributo variável:

GNAME[new index] (generic calling sequence) /va/ [index set statement] [domain statement] [range statement] [interpretation]

Para elementos do tipo função:

GNAME[new index] (generic calling sequence) /f/ [index set statement] [domain statement] generic rule statement [interpretation]

Para elementos do tipo entidade composta:

GNAME[new index] (generic calling sequence) /t/ [index set statement] [domain statement] generic rule statement [interpretation]

Simplificadamente, os elementos sintáticos de um parágrafo incluem o nome do *genus*, seu índice correspondente (se existir), o tipo do *genus*, uma declaração do conjunto do índice, especificando (ou limitando) a população do *genus*, uma declaração dos tipos permitidos do *genus* (para atributos), uma regra geral para especificar como os valores devem ser determinados (para função e teste), e uma interpretação em linguagem natural. Detalhes sobre a sintaxe dos parágrafos podem ser encontrados em [Geoffr90].

Um esquema completo para o modelo de transporte pode ser visto na Figura 2.6. A linguagem utilizada para a descrição do modelo de transporte é a SML (*Structured Modeling Language*).

&SDATA SOURCE DATA

PLANTi /pe/ There is a list of PLANTS

SUP (PLANTi) /a/ : Real+ Every PLANT has a SUPPLY CAPACITY

&CDATA CUSTOMER DATA

CUSTj /pe/ There is a list os CUSTOMERS

DEM (CUSTj) /a/ : Real+ Every CUSTOMER has a nonnegative DEMAND

&TDATA TRANSPORTATION DATA

LINK (PLANTi, CUSTj) /ce/ There is a tranportation LINK from each PLANT to each CUSTOMER.

FLOW (LINKij) /va/ : Real + Every LINK has a nonnegative transportation FLOW

COST (LINKij) /a/ Every LINK has a UNIT FLOW COST

TOTALCOST(COST, FLOW) /f/ ; @SUMi SUMj (COSTij * FLOWij) There is a TOTAL COST asociated with all FLOWS

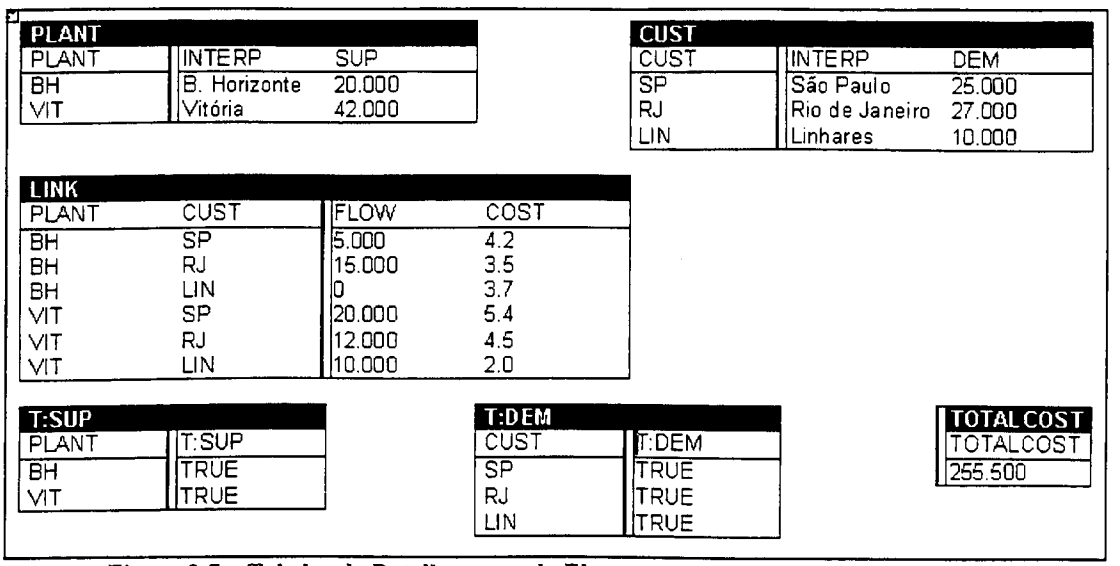
T:SUP (FLOWi., SUPi) /t/ ; @SUMj (FLOWij <= SUPi Is the total FLOW leaving a PLANT less than or equal to its SUPPLY CAPACITY? This is called the SUPPLY TEST.

T:DEM (FLOW.j, DEMj) /t/ ; @SUMi (FLOWij = DEMj Is the total FLOW arriving at a CUSTOMER exacty equal to its DEMAND? This is called the DEMAND TEST.

Figura 2.6 – Esquema do modelo de transporte em SML

2.4.4 – Tabelas de Detalhamento de Elementos

O propósito das tabelas de detalhamento de elementos é descrever uma instância particular de um esquema de modelo. A estrutura das tabelas de detalhamento de elementos é automaticamente determinada a partir do esquema do modelo seguindo algumas regras básicas [Geoffr90]. Um exemplo das tabelas de detalhamento de elementos para o modelo de transporte pode ser visto na Figura 2.7, as colunas à esquerda da linha dupla são os índices.



PLANT	INTERP	SUP
BH	B. Horizonte	20.000
VIT	Vitória	42.000

CUST	INTERP	DEM
SP	São Paulo	25.000
RJ	Rio de Janeiro	27.000
LIN	Linhares	10.000

LINK			
PLANT	CUST	FLOW	COST
BH	SP	5.000	4.2
BH	RJ	15.000	3.5
BH	LIN	0	3.7
VIT	SP	20.000	5.4
VIT	RJ	12.000	4.5
VIT	LIN	10.000	2.0

T:SUP	
PLANT	T:SUP
BH	TRUE
VIT	TRUE

T:DEM	
CUST	T:DEM
SP	TRUE
RJ	TRUE
LIN	TRUE

TOTALCOST
TOTALCOST
255.500

Figura 2.7 – Tabelas de Detalhamento de Elementos para o modelo de transporte

2.5 – Linguagem SQL-M

2.5.1 - Introdução

Um SAD deve ser fácil de usar e flexível o suficiente para permitir que os usuários manipulem os dados e os modelos de modo eficiente e eficaz, de tal forma que seja possível mudar, acrescentar, excluir, ou rearranjar elementos básicos, a fim de fornecer possibilidades de consultas para situações inesperadas (*ad-hoc*) [Turban98].

A SQL-M, que é uma extensão da linguagem SQL (Structured Query Language) [Melton93] para manipulação de modelos, foi desenvolvida visando à criação de um ambiente onde os usuários possam manipular dados e modelos de forma transparente. A linguagem SQL e os sistemas de gerenciamento de banco de dados relacionais [Elmasri94] são, atualmente, um padrão, de fato, no que se refere a bancos de dados. Como os modelos necessitam de dados para sua instanciação e posterior execução, nada melhor do que fazer com que seus gerenciamentos sejam transparentes para o usuário [Pereir97].

A extensão SQL-M é voltada para usuários de SAD. Estes usuários constantemente manipulam dados e modelos. Em SQL-M, para se trabalhar com dados, o usuário usa os comandos SQL, sem nenhuma alteração para o novo ambiente e, para se trabalhar com modelos, ele usa alguns comandos novos, seguindo o mesmo estilo e ortogonalidade da linguagem SQL. Ou seja, a extensão SQL-M enriquece a linguagem

SQL, permitindo que se manipule uma nova aplicação, sem causar nenhuma alteração no que se refere à manipulação de dados.

SQL-M é uma linguagem voltada para o gerenciamento de modelos, permitindo consulta a modelos e dados, integração entre modelos e um banco de dados, composição de modelos, e total independência modelos-dados-resolvedor. Ela não é uma linguagem de modelagem. Os modelos devem ser criados usando-se uma linguagem de definição de modelos.

A composição de modelos é um aspecto interessante de SQL-M. Cada modelo básico pode ser criado em uma linguagem de modelagem diferente e depois podem ser compostos, por meio de um simples comando SQL-M, para formar um novo modelo, de forma totalmente transparente para o usuário [Pereir97].

Vale observar que usuários de mais alto nível, como executivos, por exemplo, não usariam diretamente a linguagem SQL-M e nem mesmo SQL em seu trabalho [Pereir97]. Mas a manipulação de dados e modelos em uma única linguagem como SQL-M facilita o desenvolvimento de interfaces para esse tipo de usuário.

2.5.2 – Conceitos da SQL-M

Em SQL-M, os modelos são vistos como relações cujos atributos são suas entradas e suas saídas. Da mesma maneira que em SQL, estas relações são conhecidas como tabelas, os atributos como colunas e as tuplas como linhas. Uma tabela representando um modelo em SQL-M pode ser povoada através da execução do modelo. Nesta execução, as entradas são fornecidas, possivelmente, pela execução de uma consulta ao banco de dados, e as saídas são geradas como conseqüência da execução por um resolvedor.

Para que a SQL-M pudesse representar os mais diversos tipos de modelos, uma extensão ao subsistema de tipos foi proposta. Foram criados novos tipos de dados pré-definidos, como vetores e matrizes (representados como arranjos multidimensionais, ou unidimensionais, respectivamente). Os arranjos podem ser definidos pela palavra-chave ARRAY, e uma possível utilização seria como segue [Pereir97]:

```
CREATE TABLE ExemploArranjo
        (VETOR ARRAY [] OF INTEGER
        MATRIZ ARRAY[][] OF CHAR(5))
```

A coluna VETOR da tabela ExemploArranjo mostrada acima é uma estrutura indexada unidimensional, contendo elementos do tipo INTEGER, enquanto MATRIZ é uma estrutura indexada bi-dimensional contendo elementos do tipo CHAR(5). O número de colchetes abrindo e fechando determina a dimensão do arranjo.

A criação de modelos utilizando SQL-M, na verdade não representa a definição formal do modelo, que deve ser escrito em alguma linguagem de definição de modelos, mas a definição das informações necessárias para o gerenciamento do modelo. Devem ser definidos: o nome do modelo, suas interfaces (entradas e saídas), a chamada para a linguagem na qual o modelo foi definido, e a chamada para o resolvedor padrão a ser utilizado na sua execução. O comando utilizado para a criação de modelos em SQL-M é CREATE MODEL, que possui três cláusulas: INTERFACE, onde são definidas as entradas e saídas do modelo, MODEL, onde é feita a chamada ao modelo descrito por uma linguagem de definição de modelos, e SOLVER, onde é especificada a chamada ao resolvedor padrão do modelo. Por exemplo, o modelo de transporte, definido na seção 2.2.3 pode ser criado em SQL-M da seguinte forma:

```
CREATE MODEL TRANSP
    [INTERFACE
        (SUPPLY ARRAY [] OF NUMERIC INPUT,
        DEMAND ARRAY [] OF NUMERIC INPUT,
        LINKFLOW ARRAY [][] OF NUMERIC INPUT,
        LINKCOST ARRAY [][] OF NUMERIC INPUT,
        TOTALCOST NUMERIC OUTPUT)
    MODEL ("modelo")
    SOLVER ("resolvedor padrão")]
```

A SQL-M fornece ainda mecanismos para a instanciação do modelo através de construções próprias. A instanciação é feita através da execução do modelo que, por sua vez, é acionada pela consulta ao mesmo. Para a execução, devem ser fornecidos dados válidos para todas as entradas do modelo. As consultas são realizadas através da cláusula SELECT (semelhante à SQL). Os atributos de saída são colocados na cláusula SELECT e o nome do modelo na cláusula FROM. A nova cláusula SET, adicionada pela SQL-M, permite a atribuição de dados às entradas dos modelos. Deste modo, supondo as tabelas de detalhamento de elementos mostradas na Figura 2.7 como um

banco de dados povoado, uma consulta (e execução) ao modelo de transporte pode ser realizada da seguinte maneira:

```
SELECT TOTALCOST
FROM TRANSP
SET   SUPPLY = (SELECT Plant, Sup FROM PLANT)
      DEMAND = (SELECT Cust, Dem FROM CUST)
      LINKFLOW = (SELECT Plant, Cust, Flow FROM LINK)
      LINKCOST = (SELECT Plant, Cust, Cost FROM LINK)
```

Como pode ser observado acima, uma instância de modelo pode ser criada facilmente através das cláusulas SQL-M. O usuário pode ainda criar diferentes cenários simplesmente adicionando cláusulas WHERE nas cláusulas SELECT, restringindo assim os dados apenas aqueles que satisfaçam um determinado critério de seleção.

A SQL-M ainda fornece possibilidade de composição de modelos. Esta composição pode ser definida de duas maneiras. A primeira é por meio de uma consulta, utilizando a cláusula SELECT. A segunda é pela criação de um novo modelo, usando a cláusula CREATE MODEL. Enquanto a primeira forma não gera um novo esquema de modelo (o que permite realizar vários testes até se chegar a um modelo final), a segunda forma cria um novo modelo que pode ser armazenado na base de modelos. Exemplos de composições de modelos das duas formas podem ser vistos em [Pereir97].

2.6 – Trabalhos Existentes.

Entre os SGM existentes atualmente, os que mais se destacam são SYMMS [Muhann94], GBMS/SM [Chari98], Networks/SM [Jones92], IGOR [Hamach93] e MODASS [Gaglia96]. Apesar de nem todos os sistemas citados utilizarem a Modelagem Estruturada como base de representação dos modelos, foram incluídos por trazerem uma abordagem diferente para a Gerência de Modelos. Uma das deficiências de alguns dos sistemas citados é que geralmente eles são específicos para um determinado domínio, ou então suportam apenas uma parte do ciclo de vida de modelagem [Chari98]. Uma outra desvantagem destes sistemas é a necessidade de considerável conhecimento da sintaxe de linguagens proprietárias de modelagem e de

consulta, e a falta de um componente cooperativo para auxiliar os usuários inexperientes no processo de modelagem.

No SYMMS, os dados podem ser obtidos de um banco de dados diretamente para as entradas dos modelos, usando uma linguagem própria chamada MDL. Porém, a linguagem MDL não permite seleções específicas de uma tabela em um banco de dados relacional, permitindo apenas seleções da tabela inteira. Para seleções específicas, o usuário deve invocar um banco de dados, criar uma nova tabela com os dados desejados, voltar ao ambiente SYMMS e fornecer a nova tabela como entrada para o modelo.

O Networks/SM é um sistema baseado em gramáticas de grafos. Os modelos são representados por grafos utilizando uma interface gráfica. Porém, o Networks/SM não possui ligações externas com resolvedores, planilhas, e bancos de dados. Também não possui facilidades de indexação, nem suporte para linguagens textuais como SML [Geoffr87], GAMS [Brooke88] e AMPL [Fourer90].

IGOR (*Integrated Graphics for Operations Research*) é um sistema de gerenciamento de modelos baseado em modelagem estruturada. Utiliza conceitos de modelagem semântica de dados e suporta primariamente formulação de modelos. Também é capaz de produzir esquemas de modelo em SML [Hamach93].

MODASS é um sistema de modelagem orientada a objetos que utiliza uma linguagem orientada a objetos, chamada BLOMMS [Gaglia97], para especificar um modelo. Embora o MODASS seja geral em seu escopo, sua capacidade de geração de expressões fica limitada pela BLOMMS.

GBMS/SM (*Graph Based Modeling System for Structured* Modeling) é um ambiente gráfico de modelagem baseado em modelagem estruturada que utiliza grafos de modelo, que são grafos direcionados similares aos grafos de *genus* [Geoffr87] da Modelagem Estruturada, para a representação gráfica dos modelos. Possui ligações externas para planilhas Quattro-Pro e banco de dados Paradox. Utiliza como forma de representação interna dos modelos uma linguagem chamada MEL (*Model Expression Language*). Apesar das construções das expressões em MEL serem facilitadas pelo uso de *templates* durante a formulação do modelo, o usuário deve conhecer muito bem a sua sintaxe para conseguir uma perfeita utilização do sistema. O GBMS possui ligações para dois resolvedores, um interno (MEL) e outro externo (GAMS).

Além dos sistemas citados, as linguagens algébricas de modelagem largamente utilizadas, como GAMS [Brooke88] e AMPL [Fourer90], apesar de possuírem uma

sintaxe amigável, não permitem o uso de construções gráficas para representar os modelos. Outra restrição destas linguagens é o fato de serem utilizadas principalmente para programação matemática [Chari98].

3 – Ambiente Gráfico para Gerência de Modelos

Atualmente, existem diversos sistemas e linguagens de modelagem que utilizam abordagens gráficas e não gráficas para a representação de modelos. Os sistemas e linguagens não gráficas dificultam sua utilização, pelos gerentes, devido ao fato dos modelos resultantes se tornarem intimidadores [Dolk88]. Por sua vez, os sistemas gráficos, apesar de possuírem uma interface mais amigável para o gerente, possuem algumas deficiências que também dificultam sua utilização. Entre estas deficiências, está o fato de a maioria dos sistemas gráficos existentes para gerência de modelos ser específicos para um domínio, ou então suportar apenas parte do ciclo de vida de modelagem. Por exemplo, o IGOR [Hamacher93] suporta basicamente a formulação de modelos. O Networks/SM [Jones92], apesar de fornecer suporte para formulação, manutenção e execução de modelos, não suporta facilidades de indexação, nem ligações externas com resolvedores, bancos de dados e planilhas [Chari97a]. Além de todas estas deficiências, ainda existe um outro complicador que é a necessidade do usuário conhecer profundamente a sintaxe de uma linguagem de definição de modelos, que em alguns casos pode ser exclusiva de algum determinado sistema. Mesmo quando o sistema utiliza uma linguagem de maior poder de expressão, como a SML, ainda pode utilizar uma forma interna de representação em outra linguagem, como o caso do GBMS/SM que utiliza a MEL para representar internamente seus modelos.

Outro problema que deve ser tratado nos SGM atuais é sua interação com bancos de dados. Como os modelos necessitam dos dados para serem executados, devem ser fornecidos mecanismos eficientes para que os dados possam ser acessados da maneira mais fácil e transparente possível para o usuário. Por exemplo, no sistema SYMMS, o

usuário, além de precisar aprender uma linguagem própria (MDL), ele não permite seleções em uma tabela no banco de dados, exigindo que o usuário sempre utilize a tabela inteira, ou então ele próprio crie uma nova tabela com os dados desejados, neste caso, fora do ambiente do SYMMS. Já o GBMS/SM, que aceita que seleções sejam feitas em tabelas de bancos de dados para a instanciação dos modelos, associa esta seleção à própria especificação da classe do modelo, ou seja, para se criar uma nova instância para um modelo, é necessário alterar os dados relacionados à especificação do modelo, o que não permite a existência de várias instâncias independentes simultaneamente na base de modelos.

Como pode ser observado, através dos exemplos citados, os principais interessados na utilização de Sistemas de Apoio à Decisão que, neste caso, são os gerentes, encontram muita dificuldade em utilizar os SGM atuais, sejam eles gráficos ou não gráficos, devido às principais deficiências apresentadas.

Desta constatação, surgiu a motivação para fornecer uma ferramenta de gerência de modelos que preencha esta lacuna deixada pelos sistemas atuais, facilitando sua utilização pelos gerentes.

3.1 – Solução Visualizada

Com o objetivo de fornecer uma ferramenta de gerência de modelos aos gerentes, é necessário um ambiente onde as principais deficiências, apresentadas na seção anterior, sejam eliminadas ou contornadas. Através da observação dos sistemas existentes, algumas conclusões podem ser retiradas para guiar o projeto de uma ferramenta com esta finalidade:

- **Abordagem gráfica** – em uma ferramenta desenvolvida, tendo como usuários finais os gerentes de uma organização, é desejável que utilize uma abordagem gráfica para a formulação dos modelos e para o acesso aos dados. Através da utilização de uma interface gráfica, a capacidade de aprendizado dos usuários novatos aumenta consideravelmente, como já foi comprovado através de dados experimentais [Turban98]. Além da melhoria no aprendizado, os usuários novatos também se beneficiam em vários outros quesitos de performance como: maior rapidez e melhor qualidade no trabalho, alta produtividade e menor fadiga. Outro ponto importante, visando a utilização dos gerentes, é que, através da interface

gráfica, a necessidade de conhecimento profundo da sintaxe de uma linguagem de definição de modelos, ou de uma linguagem de consulta a dados fica bastante reduzida, permitindo, deste modo, que os gerentes possam utilizá-la com segurança.

- **Poder de expressão** – Nem todas as necessidades de decisão de uma organização podem ser atribuídas a apenas uma determinada área de conhecimento. Portanto, um SGM deve possuir uma grande capacidade de expressão, de modo que modelos das mais diferentes áreas de conhecimento possam ser representados.

- **Uniformidade de tratamento** – Não é desejável em um SGM que os dados e modelos sejam tratados de maneira diferente. Portanto, a ferramenta deve fornecer mecanismos para que o gerente possa manipular dados e modelos de maneira uniforme.

- **Suporte ao ciclo de vida de modelagem** – A ferramenta deve ser capaz de definir, consultar, executar e gerenciar os modelos criados.

- **Independência entre modelos, dados e resolvedores** – os gerentes devem ser capazes de criar diferentes instâncias, a partir de uma classe de modelos e cada instância pode ser executada utilizando diferentes resolvedores. Deste modo, a ferramenta consegue a flexibilidade necessária para as necessidades dos gerentes dentro das organizações.

- **Auxílio ao usuário** – Como o processo de modelagem é complexo, a presença de alguma forma de auxílio ao usuário durante a utilização da ferramenta é bastante desejável e necessária. Tal auxílio pode ser feito através do monitoramento das ações do usuário, impedindo ou alertando sempre que alguma anormalidade for detectada.

Para solucionar a questão do poder de expressão da ferramenta, foi necessária a escolha de uma linguagem de definição de modelos. Dentre as alternativas possíveis, foi

escolhida a SML (*Structured Modeling Language*) devido ao seu grande poder de expressão e algumas outras características. Além de ser uma linguagem formal, possui formas de representação gráficas próprias para os modelos, através dos grafos de genus, e de acordo com sua proposta, vem se tornando uma *linqua franca* para os SGM, uma vez que já existem diversos sistemas funcionando baseados em Modelagem Estruturada. A esperança é que, no futuro, os diversos sistemas possam interagir utilizando a SML como linguagem comum.

Quanto à uniformidade de tratamento, a solução encontrada foi a utilização da linguagem SQL-M como linguagem de manipulação de modelos. Como já foi detalhado na seção 2.5, a SQL-M é uma extensão da linguagem SQL para a manipulação de modelos. Com a utilização da SML e da SQL-M, esta ferramenta já possui condições para suportar todo o ciclo de vida da modelagem pois, através da SML, os modelos podem ser definidos e, através da SQL-M, os modelos podem ser gerenciados, consultados e executados.

Tanto a SML, quanto a SQL-M oferecem suporte para a independência entre os modelos, dados e resolvedores. Na ferramenta, o modelo é criado utilizando SML, e as instâncias são criadas utilizando SQL-M. A mesma classe de modelo, criada em SML, pode possuir inúmeras instâncias com diferentes dados através da SQL-M. Da mesma maneira, cada instância criada pode ser executada em mais de um resolvedor.

Como o objetivo de tal ferramenta é a sua utilização pelos gerentes, tanto a SML quanto a SQL-M não devem ser utilizadas diretamente, ou seja, não é desejável que os futuros usuários tenham que conhecer profundamente a sintaxe de uma ou de outra para conseguir utilizar a ferramenta. Assim, devem ser desenvolvidas interfaces gráficas para que o usuário possa utilizar as duas linguagens de forma simples e visual. Desta maneira, um dos problemas identificados em sistemas de gerência de modelos pode ser suavizado, pois o modelo resultante não se tornará tão incompreensível e intimidador para o gerente, quanto o seria sem uma representação gráfica.

Além dos elementos gráficos envolvendo a ferramenta, o auxílio ao usuário é outro fator importante para o sucesso de uso pelos gerentes. Um dos tipos de auxílio identificado foi a checagem direta de sintaxe durante a formulação dos modelos. A checagem direta de sintaxe garante que os modelos formulados estejam sintaticamente corretos, eliminando assim alguns problemas que poderiam ocorrer durante a utilização de usuários menos experientes em modelagem.

Para viabilizar a proposta de tal ferramenta, foi desenvolvido um protótipo, chamado GMMS (*Graphical Model Management System*) que implementa as características citadas, de forma que o sistema resultante seja de fácil utilização por gerentes com um mínimo de treinamento.

3.2 – O GMMS

O GMMS é uma proposta de um ambiente gráfico para gerência de modelos. Permite formulação, consulta, instanciação e composição de modelos utilizando uma interface gráfica baseada na Modelagem Estruturada, através da utilização de grafos de modelos (*model graphs*), que são grafos direcionados acíclicos similares ao grafos de genus, como forma de representação gráfica para os modelos [Geoffr87]. Além de suporte a Modelagem Estruturada, o GMMS possui suporte para a SQL-M [Pereir97], que é uma extensão da linguagem SQL para sua utilização com modelos. Este suporte se dá via entrada de comandos em SQL-M ou pelo uso de uma interface gráfica desenvolvida para a SQL-M, baseada no estilo QBE (*Query By Example*) [Elmasr94], onde o usuário poderá facilmente criar instâncias dos modelos existentes.

O GMMS se diferencia dos demais sistemas pela interface mais amigável e atual que auxilia o usuário em suas tarefas. Essa interface conta com a ajuda de um componente de conhecimento que fiscaliza as operações realizadas pelo usuário e o interrompe sempre que detectar alguma decisão errada ou alguma inconsistência na seqüência de uso do GMMS. Outro diferencial do GMMS é sua maior integração com a Modelagem Estruturada. Outros sistemas que utilizam Modelagem Estruturada, como o GBMS/SM e o IGOR, utilizam apenas alguns de seus elementos para a representação dos modelos. O GBMS/SM utiliza grafos de modelo para a representação gráfica dos modelos, mas a representação interna continua utilizando uma linguagem própria chamada MEL (*Model Expression Language*). O IGOR gera esquemas dos modelos em SML, mas a representação dos modelos gráficos é feita por diagramas que juntam conceitos de Entidade e Relacionamento, com uma variação dos grafos de genus da Modelagem Estruturada. A proposta do GMMS é a utilização da Modelagem Estruturada tanto para a representação gráfica dos modelos, utilizando os grafos de modelo, quanto para a representação interna, utilizando SML, de tal maneira que dado um modelo representado graficamente, pode-se obter seu equivalente em SML através de um mapeamento da linguagem gráfica para a linguagem textual.

O GMMS também conta com suporte para a linguagem SQL-M [Pereir97], que é uma extensão da linguagem SQL para a manipulação de modelos. A SQL-M é uma linguagem voltada para o gerenciamento de modelos, permitindo consulta a modelos e dados, integração entre modelos e um banco de dados, composição de modelos, e total independência modelos-dados-resolvedor [Pereir97]. Tem uma vantagem sobre outras linguagens de consulta de modelos por ser muito simples e ter sintaxe similar aos comandos SQL. Outro ponto forte da SQL-M é a possibilidade de se criar composições de modelos, mesmo que os modelos estejam definidos em linguagens diferentes, tudo de forma transparente para o usuário. O GMMS ainda introduz um avanço na SQL-M desenvolvendo uma interface gráfica, baseada em ferramentas QBE (*Query By Example*).

Todas as informações sobre os modelos ficarão armazenadas em um banco de dados relacional para permitir a fácil utilização dos modelos por sistemas externos como planilhas, e resolvedores. Os resolvedores poderão acessar as Tabelas de Detalhamento de Elementos (*Elemental Detail Tables*) das instâncias dos modelos, ou sua tradução no formato específico do resolvedor para conseguir os dados necessários para executar o modelo. Os dados gerados pelo resolvedor devem ser retornados para o usuário do GMMS de forma simples e clara.

A arquitetura proposta para o GMMS é mostrada na Figura 3.1. Podem ser vistos os principais subsistemas que compõem o GMMS (elipses com fundo cinza) e suas interações.

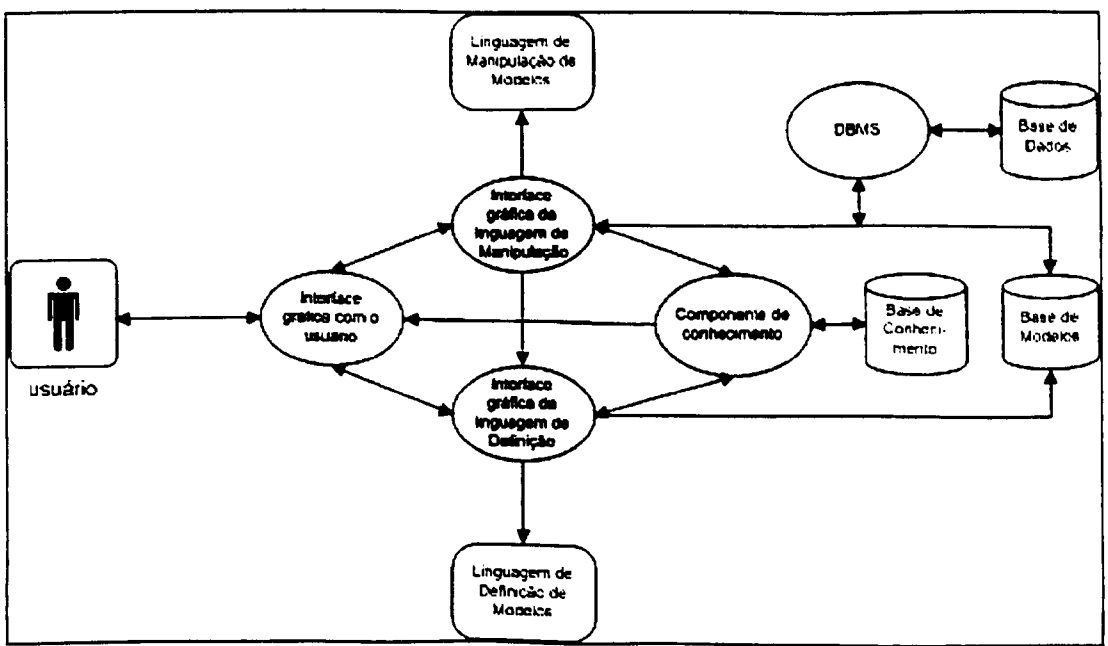


Figura 3.1 – Arquitetura do GMMS

O subsistema de Interface Gráfica com o Usuário é o responsável por todo o gerenciamento sobre o que é repassado para a visualização pelo usuário. Interage diretamente com todos os outros subsistemas, recebendo informações e exibindo-as em um formato simples e claro.

O subsistema da Interface Gráfica da Linguagem de Manipulação será o responsável pelo controle da linguagem SQL-M. Através deste subsistema, o usuário poderá criar consultas em SQL-M de maneira similar à criação de consultas SQL com auxílio de ferramentas QBE. Através de sua interação com o subsistema de Interface Gráfica da Linguagem de definição, o subsistema poderá reconhecer a estrutura do modelo em SML automaticamente deixando para o usuário, o trabalho de indicar onde estão os dados de que o modelo necessita para que uma instância seja criada. Outra característica deste subsistema é realizar a ligação do GMMS com os resolvedores. Uma grande variedade de resolvedores poderá ser suportada pelo GMMS, bastando definir uma interface de comunicação. Deste modo o GMMS pode invocar o resolvedor para um determinado modelo e retornar os valores obtidos de forma transparente para o usuário.

O subsistema da interface gráfica da linguagem de definição é o responsável pela formulação e edição dos modelos. Através deste subsistema, o usuário poderá formular modelos de maneira simples, utilizando a interface gráfica como se estivesse montando um diagrama.

O subsistema do Componente de Conhecimento é o responsável pelas informações de que o GMMS dispõe para auxiliar o usuário em suas tarefas. Através deste subsistema, o usuário terá suas ações monitoradas e sempre que o componente de conhecimento identificar um possível auxilio para o usuário, as informações pertinentes serão disponibilizadas automaticamente

Nas próximas seções será melhor explicado o funcionamento de cada um dos subsistemas do GMMS.

3.2.1 – Interface Gráfica da Linguagem de Definição de Modelos

Este subsistema é o responsável pela ligação entre a Linguagem de Definição de Modelos (que neste caso é a SML) e o GMMS. Para que o gerente possa construir um modelo graficamente, foi necessário projetar uma representação gráfica para a SML, de forma que a linguagem gráfica resultante não perdesse seu poder de expressão. Como a

modelagem estruturada já provê mecanismos de representação gráfica para os esquemas de modelos, foi feita uma adaptação para que o modelo gráfico, além de fornecer a informação sobre a estrutura do modelo, também forneça as informações mais detalhadas sobre cada elemento que compõe o modelo. Esta adaptação foi feita através de grafos de modelos, para representar a estrutura do modelo, onde cada elemento do modelo pode ter suas características fornecidas através de formulários próprios para cada tipo de elemento.

Os elementos da SML foram representados com formas geométricas simples, como mostrado na Figura 3.2, onde um quadrado representa uma entidade primitiva (pe), um retângulo representa uma entidade composta (ce), um círculo representa uma variável (a), um triangulo representa uma função (f), e um losango representa um teste (t).

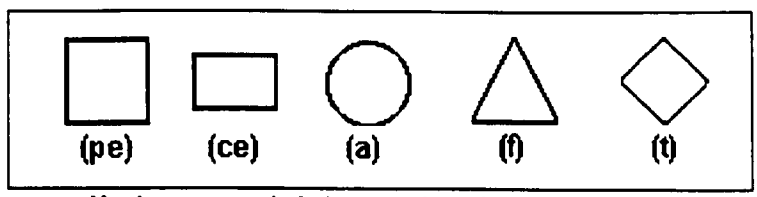


Figura 3.2 – Correspondência entre os símbolos e os tipos de elementos da Modelagem Estruturada

Para que o poder de expressão da SML fosse preservado, foi necessário associar, a cada elemento gráfico, informações suficientes para que fosse possível a criação de parágrafos em SML, de acordo com a sintaxe mostrada na seção 2.2.3.

As informações necessárias para cada tipo de elemento para gerar esquemas de modelos em SML podem ser vistas na Tabela 3.1.

Tipo de elemento	Informações necessárias
Entidade Primitiva	Nome, Tipo, Índice, Interpretação
Entidade composta	Nome, Tipo, Índice, Interpretação
Variável	Nome, Tipo, Índice, Domínio, Interpretação
Teste	Nome, Tipo, Índice, Expressão Teste, Interpretação
Função	Nome, Tipo, Índice, Expressão Função, Domínio, interpretação

Tabela 3.1 – Informações necessárias para os elementos de um modelo

Um esquema de modelo em SML, como o mostrado na Figura 3.3, pode ser representado graficamente pelo diagrama mostrado na Figura 3.4, gerado através do GMMS. As informações pertinentes a cada um dos elementos são mostradas na Figura 3.5.

SATELLITE /pe/ Existe um SATELITE no espaço.

OBJECT /pe/ Existe um OBJETO no espaço.

S_MASS (SATELLITE) /a/ **: Real+** O SATELITE possui uma certa MASSA DO SATELITE.

O_MASS (OBJECT) /va/ **: Real+** O OBJETO possui uma certa MASSA DO OBJETO.

D (SATELLITE, OBJECT) /va/ **: Real+** O SATELLITE e o OBJETO estão a uma certa DISTÂNCIA em metros.

FORCE (S_MASS, O_MASS, D) /f/ **: 6.67 * 10^(-11) * S_MASS * O_MASS / D^2** O objeto exerce uma certa FORÇA no SATELLITE, em newtons, obedecendo a Lei de gravitação universal.

THREAT (FORCE) /t/ **; FORCE > 10^(-6)** O OBJETO representa uma AMEAÇA ao satélite, se, e somente se, ele exerce uma FORÇA maior do que um milhão de newtons.

Figura 3.3 – Esquema de modelo SATELLITE [Geoffr92a]

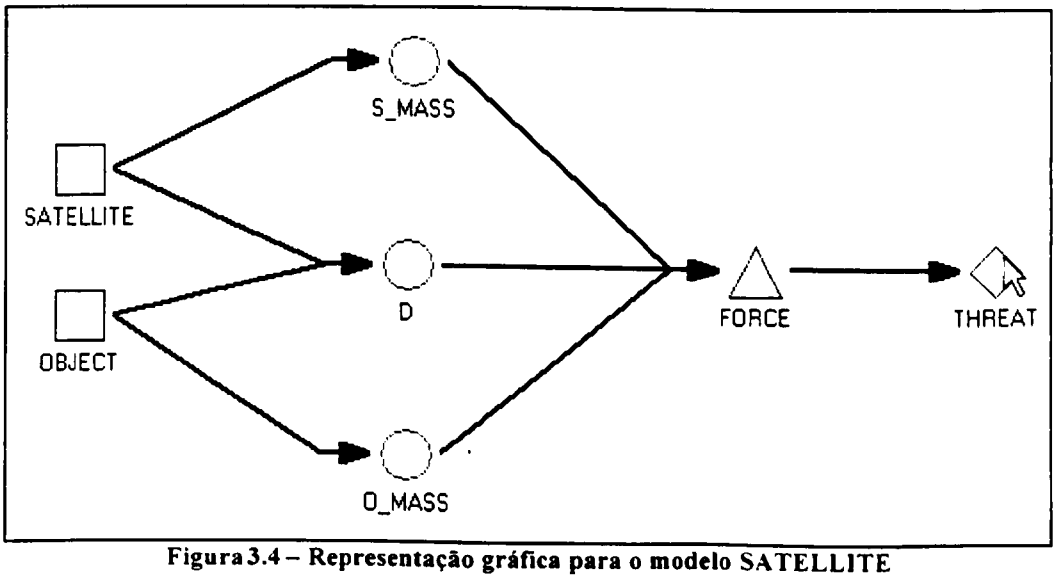


Figura 3.4 – Representação gráfica para o modelo SATELLITE

O atributo "índice" é responsável pela indexação dos elementos origem para os elementos destino. Algumas opções válidas para a indexação simples dos elementos são mostradas na Tabela 3.2; para maiores detalhes, ver [Geoffr92b].

O ponto serve para realizar substituições do tipo "para todos", enquanto a opção *<inteiro>* serve para substituir um determinado elemento na posição ordinal *inteiro*. O *<inteiro>* pode ser tanto positivo quanto negativo. O significado de inteiros negativos é que a contagem da posição ordinal do elemento deve começar de trás para frente. O intervalo definido por *<inteiro1:inteiro2>* denota um intervalo nos elementos começando em *inteiro1* e terminando em *inteiro2*. Para maiores detalhes, consultar [Geoffr92b].

SATELLITE
Tipo: pe
Interpretação: Existe um SATELITE no espaço
Índice: -

OBJECT
Tipo: pe
Interpretação: Existe um OBJETO no espaço
Índice:

S_MASS
Tipo: a
Interpretação: O SATELITE possui uma certa MASSA DO SATELITE.
Domínio: Real+
Índice: SATELLITE

O_MASS
Tipo: va
Interpretação: O OBJETO possui uma certa MASSA DO OBJETO.
Domínio: Real+
Índice: OBJECT

D
Tipo: va
Interpretação: O SATELITE e o OBJETO estão a uma certa DISTÂNCIA em metros.
Domínio: Real+
Índice: SATELLITE, OBJECT

FORCE
Tipo: f
Interpretação: O OBJETO exerce uma certa FORÇA no SATELLITE, em newtons, obedecendo a Lei de gravitação universal.
Domínio: Real+
Expressão: 6.67 * 10^(-11) * S_MASS * O_MASS / D^2
Índice: S_MASS, O_MASS, D

THREAT
Tipo: t
Interpretação: O OBJETO representa uma AMEAÇA ao satélite, se, e somente se, ele exerce uma FORÇA maior do que um milhão de newtons.
Domínio: Logical
Expressão: FORCE > 10^(-6)
Índice: D

Figura 3.5 – Informações associadas aos elementos gráficos do GMMS para o Modelo SATELLITE

GENUSij	chamada para o elemento GENUSij
GENUS.j	chamada para GENUSij para todo i
GENUSi.	chamada para GENUSij para todo j
GENUSi<2>	chamada para GENUSij para o segundo identificador j
GENUSi<1:-2>	chamada para GENUSij para todo j, exceto o último
GENUS.. ou GENUS	chamada para GENUSij para todo i e j

Tabela 3.2 – Exemplos de substituições de índices

3.2.2 – Interface Gráfica da Linguagem de Manipulação de Modelos

A interface gráfica para a linguagem de manipulação de modelos é responsável pela ligação entre a linguagem SQL-M e o GMMS. Através dela, o usuário pode criar instâncias de modelos, simplesmente fornecendo dados para suas variáveis de entrada. Também é possível executar a instância de modelo criada, ou editar alguma instância criada anteriormente. Outra responsabilidade deste subsistema é a ligação entre o GMMS e os resolvedores. As instâncias criadas possuem um resolvedor padrão, e o resolvedor é capaz de reconhecer os dados necessários para a execução da instância do modelo. A saída gerada pelo resolvedor é reconhecida pelo GMMS, que as repassa ao usuário de maneira simples e clara.

Este subsistema também possui uma ligação com o subsistema de linguagem de definição de modelos. Tal ligação deve-se ao fato de que a linguagem de manipulação deve reconhecer a interface do modelo, ou seja, as variáveis de entrada e saída bem como seus respectivos tipos de dados.

Para este reconhecimento, foram estabelecidas algumas regras que dizem respeito à classificação das variáveis (entrada ou saída), quanto à dimensão (simples, vetor, array) e quanto ao tipo de dados (real, inteiro, string, lógico):

Quanto a classificação:

- Elementos do tipo entidade primitiva e entidade composta não geram variáveis.
- Cada elemento do tipo atributo, ou atributo variável se converterá em uma variável de entrada do modelo.
- Cada elemento do tipo função, ou teste se converterá em uma variável de saída do modelo.

Quanto a dimensão:

- Elementos que não possuem nenhum elemento em sua seqüência de chamada não geram variáveis.
- Elementos que possuem um, ou mais elementos na seqüência de chamada:
    - Neste caso, para determinar a dimensão, basta verificar a Tupla Genérica de Índices [Geoffr90] do *genus*. O número de índices encontrado será o número de dimensões do array.

Quanto ao tipo de dados:

- Integer ou Integer+ ou Integer- ou Int→ interpretados como INTEGER em SQL-M.
- Real ou Real+ ou Real- → interpretados como NUMERIC em SQL-M.
- String ou String *n* → interpretados como Char, ou Char(*n*) respectivamente.
- Lógico (testes) → interpretados como BOOLEAN em SQL-M.

Seguindo estas regras, a criação do modelo SATELLITE em SQL-M fica como segue:

```
CREATE MODEL SATELLITE
    [INTERFACE
        (S_MASS NUMERIC INPUT,
        O_MASS NUMERIC INPUT,
        D NUMERIC INPUT,
        FORCE NUMERIC OUTPUT,
        THREAT BOOLEAN OUTPUT)]
    MODEL ("satellite")
    SOLVER ("satellite generic solver")]
```

Deste modo, o trabalho necessário para o usuário criar uma instância completa do modelo é: fornecer o caminho, quais dados devem ser utilizados, especificar o resolvedor e indicar qual é a classe de modelo sendo instanciada. O fornecimento dos dados é realizado de maneira visual, utilizando ferramentas para a construção de consultas SQL, sem que seja necessário que o usuário conheça profundamente sua sintaxe. Ele deve simplesmente fornecer a tabela onde os dados se encontram, selecionar os campos necessários e fornecer algum critério de seleção caso seja necessário. Este processo deve ser repetido para cada variável de entrada que o modelo possuir. Assim, cada variável terá uma consulta SQL associada, indicando os dados necessários para a criação da instância do modelo.

Diferentemente do GBMS/SM, onde também é possível associar consultas SQL para a obtenção dos dados, o GMMS não associa as consultas SQL à classe de modelos criada, podendo existir várias instâncias do mesmo modelo utilizando consultas SQL diferentes, enquanto, no GBMS/SM, seria necessário sempre modificar as consultas

SQL para criar uma nova instância (perdendo a antiga) ou, então, copiar o modelo inteiro e associar novas consultas SQL.

Supondo a existência de uma tabela com a estrutura mostrada na Tabela 3.3, após a instanciação do modelo SATELLITE, a seguinte consulta SQL-M pode ser gerada pelo GMMS:

```
SELECT FORCE, THREAT
FROM SATELLITE
SET   S_MASS = (SELECT MassaSatelite FROM Satelites)
      O_MASS = (SELECT MassaObjeto FROM Satelites)
      D = (SELECT Distancia FROM Satelites)
```

Satelite	Objeto	MassaSatelite	MassaObjeto	Distancia
DPI-Sat5	Melkor-x3	100	2000	200

Tabela 3.3 – Tabela Satelites com os dados para uma instância do modelo SATELLITE

Com a consulta pronta, a instância pode ser enviada para a solução em algum resolvedor. Mas, diferente do que acontece na implementação proposta por [Pereir97], onde o resultado da execução era retornado como uma tabela contendo uma única linha, e onde as colunas eram as variáveis selecionadas pela instância (FORCE e THREAT, no caso do exemplo do SATELLITE), no GMMS os resultados do resolvedor são direcionados para as respectivas variáveis de saída. Deste modo, quando o usuário desejar verificar o resultado de alguma execução, basta selecionar a variável desejada que o seu valor atual será exibido. Assim, o resultado gerado pelo resolvedor pode ser consultado mais facilmente pelo gerente.

3.2.3 – Componente de Conhecimento

O subsistema do componente de conhecimento centraliza todas as informações necessárias para o auxílio que o GMMS pode dispensar ao usuário. Atua diretamente nos subsistemas de linguagem de definição e de manipulação. Sua atuação consiste em uma série de verificações nas ações do usuário e avisos de possíveis erros que este possa cometer.

Sua atuação no subsistema de linguagem de definição é basicamente fornecer as regras para a checagem direta de sintaxe durante a construção do modelo. Seguindo os

princípios da Modelagem Estruturada, as ligações entre os nodos no grafo de modelo devem respeitar algumas restrições. Classificando os nodos como nodo origem ou nodo destino, as restrições de ligações entre os nodos são listadas na Tabela 3.4. Utilizando estes princípios, o GMMS assegura que os modelos criados graficamente são sintaticamente corretos.

Nodo Destino		Possíveis Nodos Origem
Entidade Primitiva	(pe)	Nenhum
Entidade Composta	(ce)	pe ou ce
Atributo	(a)	pe ou ce
Função	(f)	pe, ce, a, f ou t
Teste	(t)	pe, ce, a, f ou t

Tabela 3.4 - Restrições de relacionamentos de nodos nos grafos de modelos

Já no subsistema de linguagem de manipulação, o componente de conhecimento atua realizando a checagem de compatibilidade entre os tipos e dimensões de dados que o usuário fornece para os dados de entrada. Caso alguma incoerência seja detectada, o usuário é avisado do erro, e o componente de conhecimento indica a possível solução.

Outra função do componente de conhecimento é gerenciar o Assistente de nova instância, que é uma seqüência de passos, guiada pelo GMMS e acompanhada pelo usuário, onde o produto final é uma instância de modelo completa. O Assistente entra em funcionamento quando o usuário desejar criar uma nova instância, a partir de alguma classe de modelo existente. Possui três etapas distintas:

1. Definição das características da Instância, como nome, descrição e resolvedor que deverá ser utilizado para a execução.
2. Identificação da interface do modelo, selecionando as variáveis entre entrada e saída.
3. Definição da fonte dos dados das variáveis de entrada.

Ao final de sua execução, o GMMS já pode exibir a instância do modelo na tela apropriada, onde o usuário pode ainda realizar alguma modificação, executar o modelo, ou simplesmente salvar a instância criada para utilização futura.

3.2.4 – Interface Gráfica do Usuário

O subsistema de interface gráfica com o usuário integra os demais subsistemas para que GMMS tenha a coesão necessária, no intuito de tornar todos os processos e tarefas realizadas transparente para o usuário. Todo o controle de ações disponíveis que o usuário pode realizar são controlados pela interface gráfica, que funciona através de menus, menus *pull-down*, barras de ferramentas, ícones, caixas de listagem, mouse, e outros conhecidos recursos gráficos do ambiente Windows.

Assim que o GMMS é iniciado, a tela inicial é mostrada, e as possíveis opções de utilização estão acessíveis através de menus, e da barra de ferramentas, localizados na parte superior da tela, como mostra a Figura 3.6.

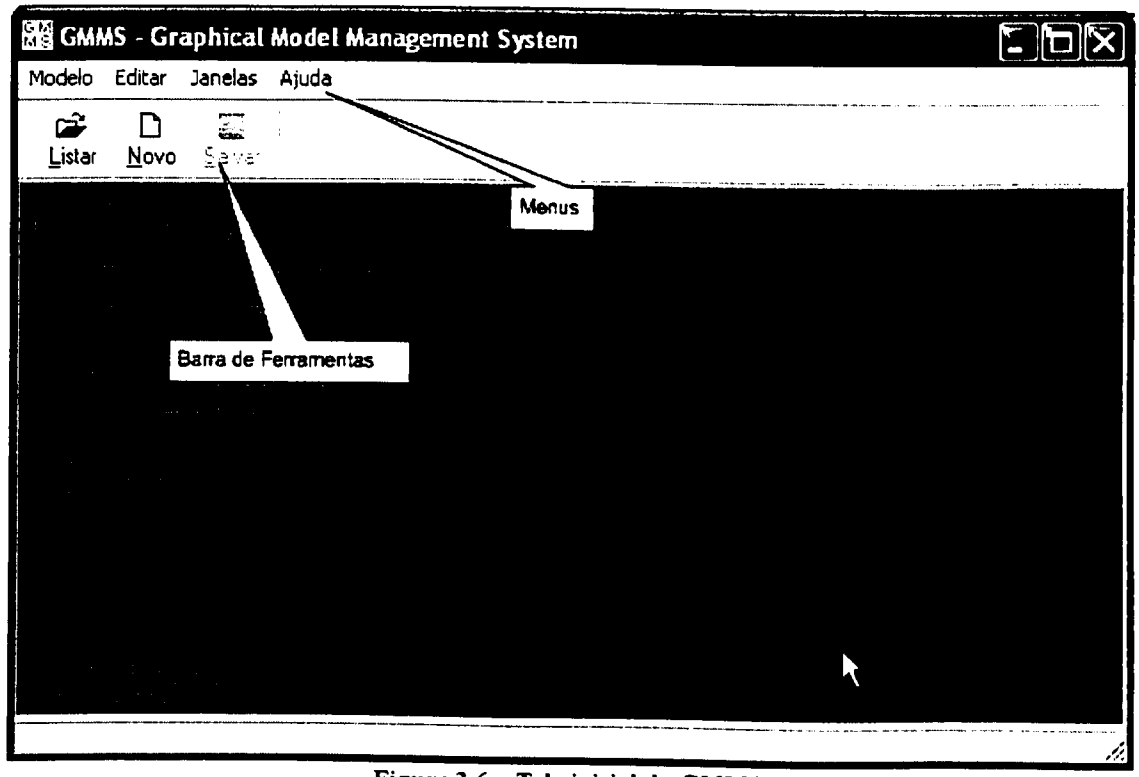


Figura 3.6 – Tela inicial do GMMS

As opções da barra de ferramenta da tela inicial, seguidas de uma breve explicação, são mostradas na Figura 3.7.

Caso o usuário selecione a primeira opção da barra de ferramentas, a tela listar é exibida, como mostra a Figura 3.8. Note que a listagem de modelos existentes, na parte superior da tela possui um registro (FEEDMIX). Do mesmo modo, a listagem de instâncias existentes também possui um registro (Ração Bovina).

 Listar	Abre a tela de listagem dos modelos e instâncias, que será detalhada em breve
 Novo	Abre a tela para formulação gráfica de uma nova classe de modelo.
 Salvar	Salva o modelo atual, só disponível quando algum modelo está aberto

Figura 3.7 – Opções disponíveis na Barra de Ferramentas da Tela Inicial

Caso a opção escolhida seja a segunda, a tela de edição gráfica do novo modelo será aberta (Figura 3.9), e o usuário poderá então formular visualmente um modelo utilizando a barra de ferramentas, posicionada na parte superior da tela, onde estão posicionados os botões com os ícones correspondentes aos tipos de elementos existentes em um esquema em SML. O GMMS utiliza símbolos geométricos simples para representar cada um dos elementos, além de uma cor diferente para cada um, com o objetivo de melhorar a identificação visual de cada elemento dentro do esquema de modelo. A barra de ferramentas da tela de edição gráfica de modelos possui dois componentes: uma barra de ferramentas SML, e outra barra de ferramentas do diagrama. Ambas podem ser vistas na Figura 3.10, assim como uma breve descrição de cada botão.

Na inserção de um nodo no diagrama, o GMMS abre uma tela específica, de acordo com o tipo de nodo. Deste modo, antes que o nodo seja inserido no diagrama, o usuário deve preencher as informações referentes ao nodo. O tipo de informação que o usuário deve informar é o mesmo do indicado na Figura 3.5.

As telas de propriedades dos tipos de nodos pe, ce, a, f, t podem ser vistas nas Figura 3.11 (a), (b), (c), (d), e (e) respectivamente.

Note que as telas de propriedades dos nodos, com exceção, dos nodos tipo pe, possuem uma listagem dos possíveis nodo origem, do lado inferior esquerdo da tela. Caso o usuário selecione nodos origem, para o nodo sendo inserido, assim que a tela de propriedades de fechar, o nodo aparecerá no diagrama já com as conexões estabelecidas. Esta maneira de inserir os nodos incentiva a abordagem *top-down* para a formulação do modelo, uma vez que a listagem de possíveis nodos origem necessita de elementos de nível mais alto que o nodo sendo inserido, além de acelerar o processo de criação dos modelos, eliminando o processo manual de conexão entre os nodos. Apesar de adotar

uma semântica de abordagem *top-down*, diferente daquela apresentada por Geoffrion [Geoffr87] para modelagem, o modelo adotado no GMMS mostrou-se eficiente durante sua utilização.

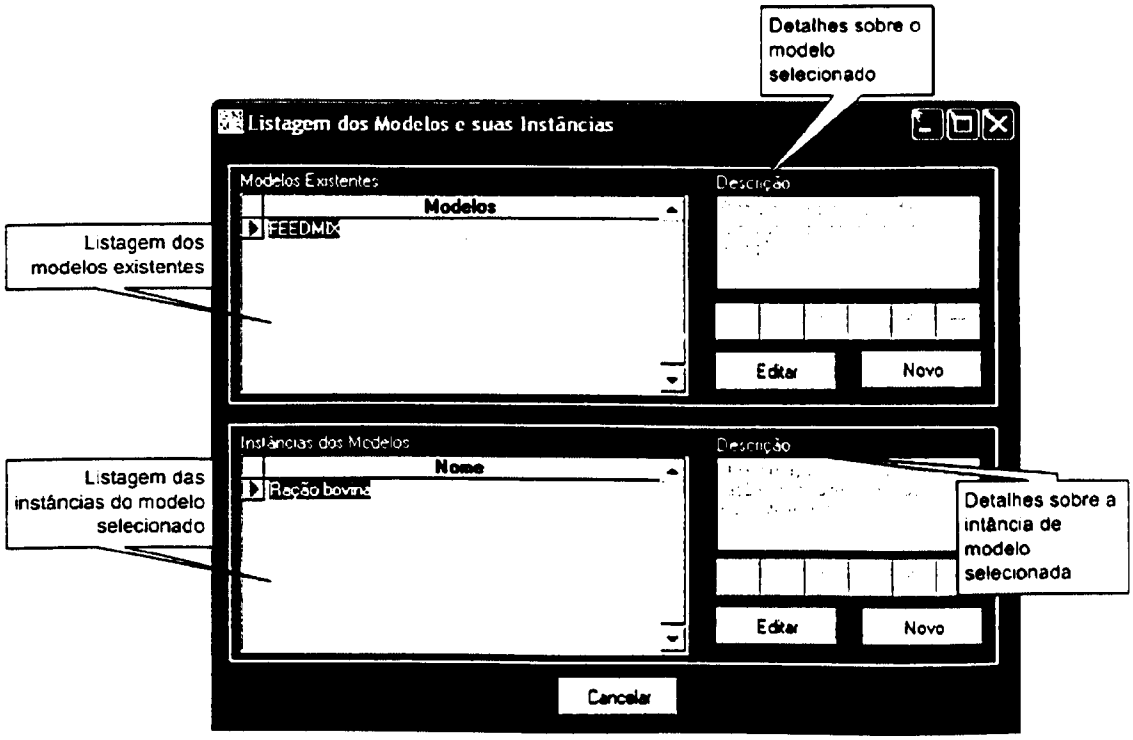


Figura 3.8 – Tela de listagem dos Modelos e Instâncias

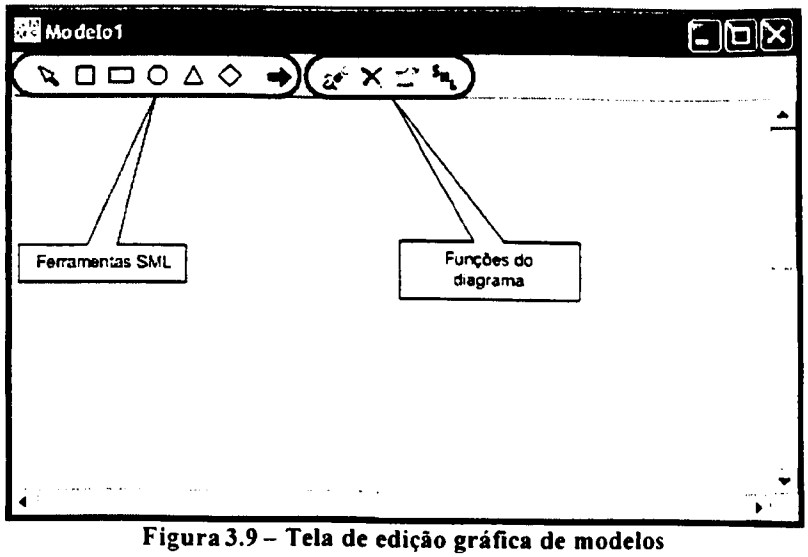


Figura 3.9 – Tela de edição gráfica de modelos

Ferramentas SML

 **Ferramenta de seleção**. Permite selecionar objetos no diagrama.

 **Inserir pe**. Insere um nodo do tipo Entidade Primitiva

 **Inserir ce**. Insere um nodo do tipo Entidade Composta

 **Inserir a**. Insere um nodo do tipo Atributo, ou Atributo Variável

 **Inserir f**. Insere um nodo do tipo Função.

 **Inserir t**. Insere um nodo do tipo Teste

 **Conectar**. Permite conectar dois nodos no diagrama

Funções do Diagrama

 **Limpar**. Permite apagar todo o conteúdo do diagrama de uma só vez

 **Propriedades**. Abre a tela com a propriedades do nodo selecionado.

 **Apagar**. Apaga o nodo selecionado.

 **Gerar SML**. Gera um esquema textual em SML a partir do modelo gráfico.

Figura 3.10 – Componentes da barra de ferramentas de criação de modelos

Através da utilização dos tipos de nodos e das conexões, um novo modelo pode ser criado no ambiente do GMMS de maneira simples, como será mostrado na seção 3.3.

A tela de edição gráfica de modelos também pode ser acessada a partir da tela de listagem de modelos, utilizando o botão "Novo" posicionado ao lado direito da listagem de modelos.

Para a criação de uma nova instância de modelo, basta acionar o botão "Novo", posicionado ao lado da listagem de instâncias, na tela de listagem de modelos. Uma vez acionado, o GMMS apresentará ao usuário um assistente para a criação de uma nova instância de modelo. As três etapas do assistente, como definidas na seção 3.2.3, podem ser vistas na Figura 3.12.

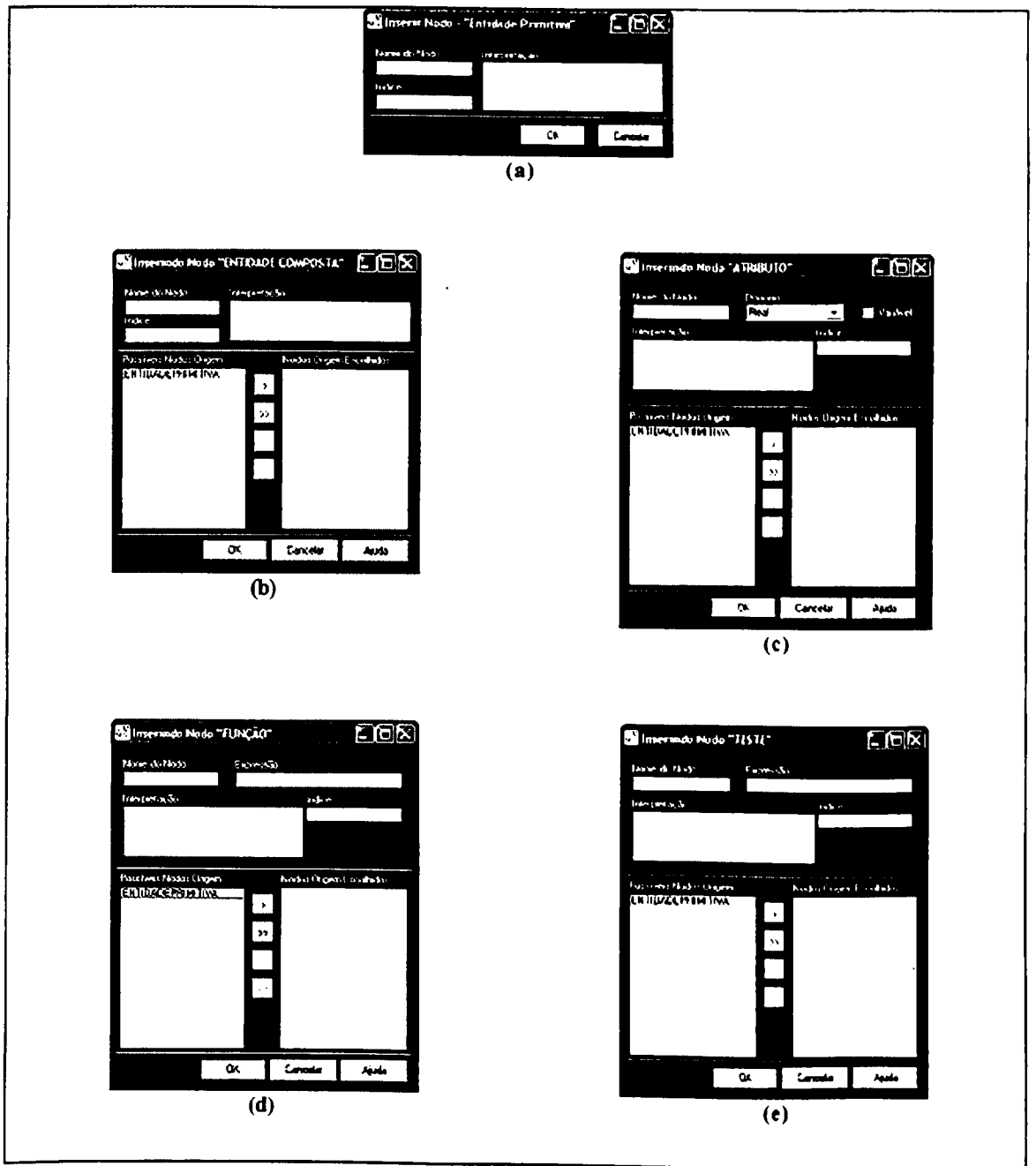


Figura 3.11 – Telas de propriedades dos nodos – (a) Inserir nodo pe; (b) Inserir nodo ce; (c) Inserir nodo atributo; (d) Inserir nodo função; (e) Inserir nodo teste.

Durante a execução do assistente, mais precisamente na terceira etapa (Figura 3.12) , deverão ser fornecidos dados para as variáveis de entrada do modelo. Para realizar esta tarefa, existem duas opções válidas. A primeira é fornecer os dados diretamente. A segunda é fornecer uma consulta SQL, indicando onde e quais os dados devem ser utilizados. Caso a segunda opção seja utilizada, o GMMS apresentará a tela de criação visual de consultas SQL (Figura 3.13), onde as consultas podem ser facilmente criadas de maneira visual, semelhante as ferramentas QBE. Assim que uma

consulta for criada, ela é repassada ao assistente que deverá verificar se ela é válida (os tipos de dados da consulta, e da variável devem ser compatíveis).

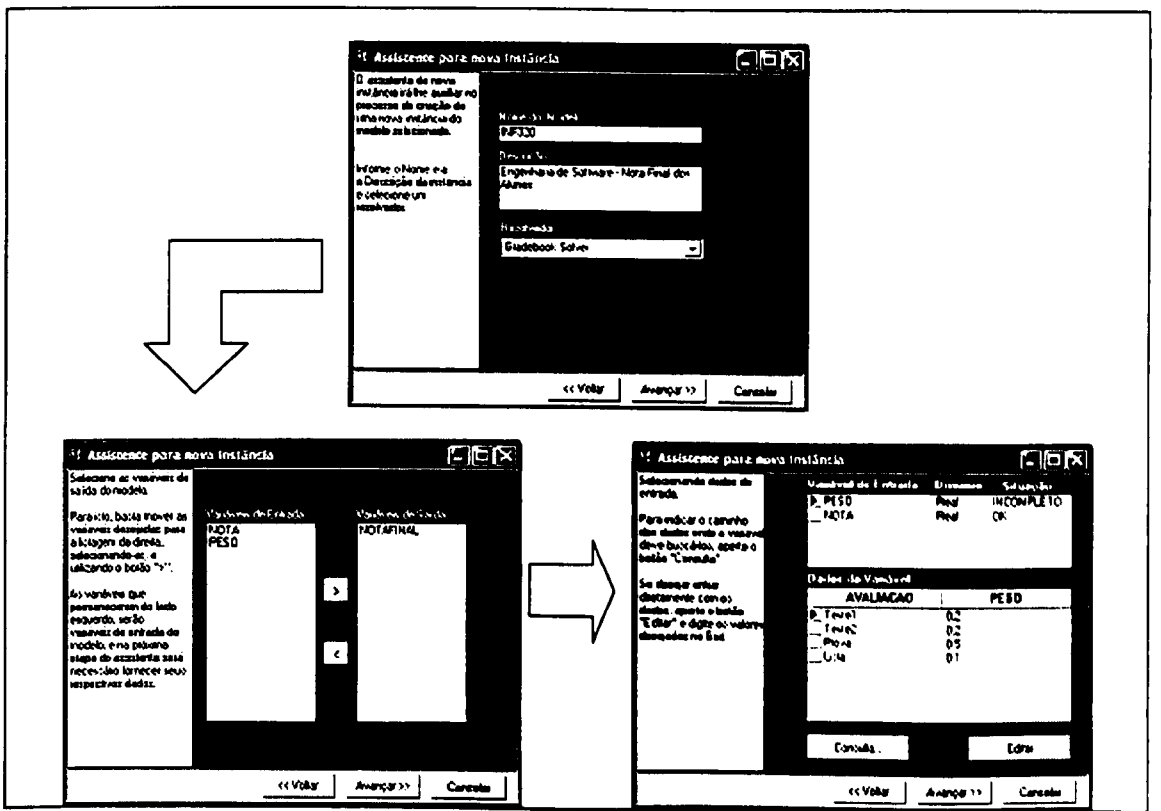


Figura 3.12 – Etapas do assistente para nova instância

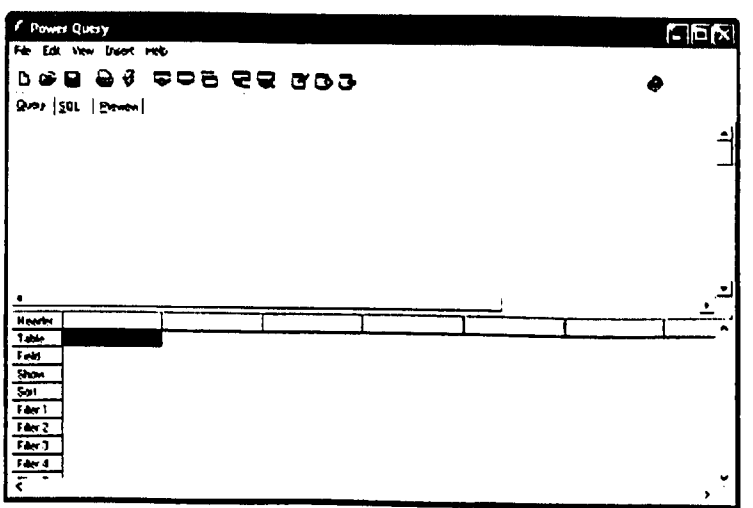


Figura 3.13 – Construção visual de consultas SQL

O assistente termina sua execução assim que as variáveis de entrada possuírem dados válidos. O GMMS então, exibe a tela de edição de instância. Nesta tela, é possível visualizar os dados preenchidos durante a execução do assistente, modificar

estes dados, ou então executar a instância utilizando o resolvedor que foi fornecido na primeira etapa do assistente.

Ao executar a instância, o GMMS se comunica com o resolvedor, passando o modelo e os dados instanciados, o resolvedor executa os cálculos necessários e devolve as soluções encontradas para o GMMS, que se encarrega de instanciar as variáveis de saída com os dados recebidos. A tela de edição de instância pode ser vista na Figura 3.14.

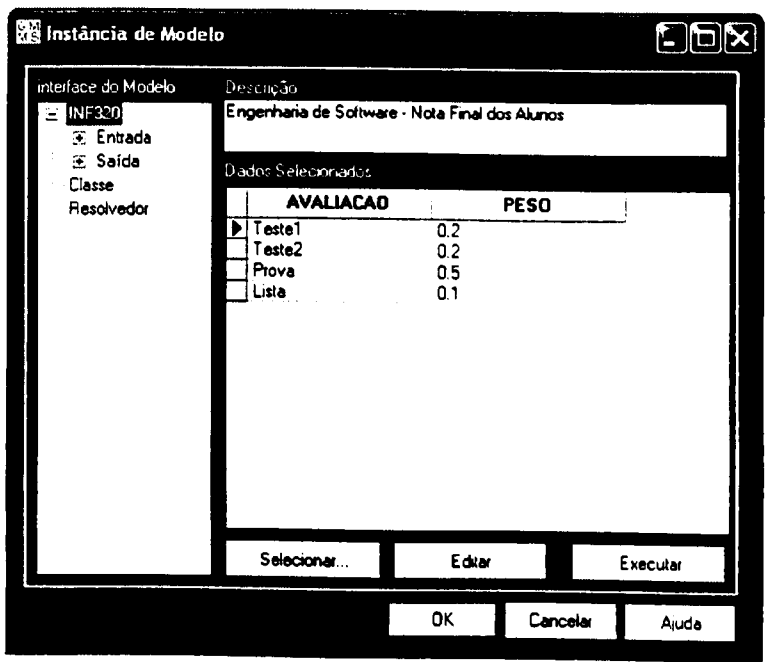


Figura 3.14 – Tela de edição das instâncias

3.3 – Cenário de Uso

Para demonstrar a utilização do GMMS, será utilizado mais uma vez o modelo de transporte introduzido na seção 2.2.3. Naquela ocasião, foi mostrado como a SML poderia representar o modelo de transporte. Agora será mostrado como o GMMS utiliza sua interface gráfica para representar o mesmo problema, descrito em SML de maneira mais simples de ser entendida por usuários não especialistas. A situação que será modelada é a mesma descrita na seção 2.2.3.

Ao utilizar o GMMS para resolver este problema de modelagem, pode ser definida uma seqüência de ações que o usuário deverá efetuar em situações deste tipo, chegando a uma estrutura de atividades como mostrada pelo diagrama da Figura 3.15.

Através do diagrama, pode-se facilmente perceber o papel de cada um dos atores envolvidos no sistema, como o usuário, o GMMS e os resolvedores.

Seguindo a ordem sugerida, o usuário do GMMS primeiramente deve acessar a tela de listagem de modelos e suas instâncias (Figura 3.8) para verificar a existência de um modelo que resolva o problema. Como nesta primeira etapa, o modelo não se encontra na base de modelos, será necessário criar o modelo, utilizando a tela de edição gráfica do modelo (Figura 3.9).

Utilizando uma abordagem *top-down*, podem ser definidas inicialmente as entidades de mais alto nível. Examinando a descrição do problema, percebe-se dois elementos cujas definições são independentes dos demais: *Fábrica* e *Consumidor*.

Outros elementos, que são: *Ligações*, *Fornecimento*, *Demanda*, *Custo*, e *Fluxo* possuem sua definição dependente diretamente de *Fábrica*, ou de *Consumidor*. O *Teste de Fornecimento* e o *Teste de Demanda* utilizam os elementos *Fornecimento* e *Demanda*, respectivamente em suas definições, enquanto o *Custo Total* do transporte é calculado baseado nas informações dos elementos *Fluxo* e *Custo*.

Na Figura 3.16a, um elemento do tipo entidade primitiva, representando as fábricas, foi inserido no diagrama, como mostra a Figura 3.16b. Como está sendo considerada uma lista de fábricas, devemos indexar o elemento. Para isto, escolhemos uma variável de índice, que deve ser alguma letra em caixa baixa, neste caso utilizaremos a letra "i". Da mesma maneira, na Figura 3.17a, um outro elemento do tipo entidade primitiva foi inserido para representar os consumidores resultando no diagrama da Figura 3.17b. Semelhante às fábricas, os consumidores também constituem uma lista, logo devem ser indexados utilizando alguma outra letra, neste caso será utilizada a letra "j".

Agora que o diagrama já possui os elementos correspondentes às fabricas e aos consumidores, os elementos que dependem da definição destes dois elementos já podem ser inseridos. Começando pelo elemento entidade composta *Ligação* (Figura 3.18a). Note que os elementos *Fábrica* e *Consumidores* aparecem na listagem dos possíveis nodo origem. Selecionando os dois na listagem, já que na definição de *Ligação* os dois elementos são utilizados, percebe-se que no diagrama os elementos já aparecem com suas respectivas conexões estabelecidas (Figura 3.18b). Apesar do elemento *Ligação* não possuir um índice próprio, ele possui um índice externo, proveniente dos nodos aos quais ele se relaciona (no caso *Fábrica* e *Consumidores*). Em razão do índice externo, deve ser indicado no elemento *Ligação*, quais elementos dos nodos relacionados

deverão ser considerados. Como uma *Ligação* deve existir entre cada um dos elementos *Fábrica* e *Consumidor*, o índice de *Ligação* será "FABRICAi. CONSUMIDORESj".

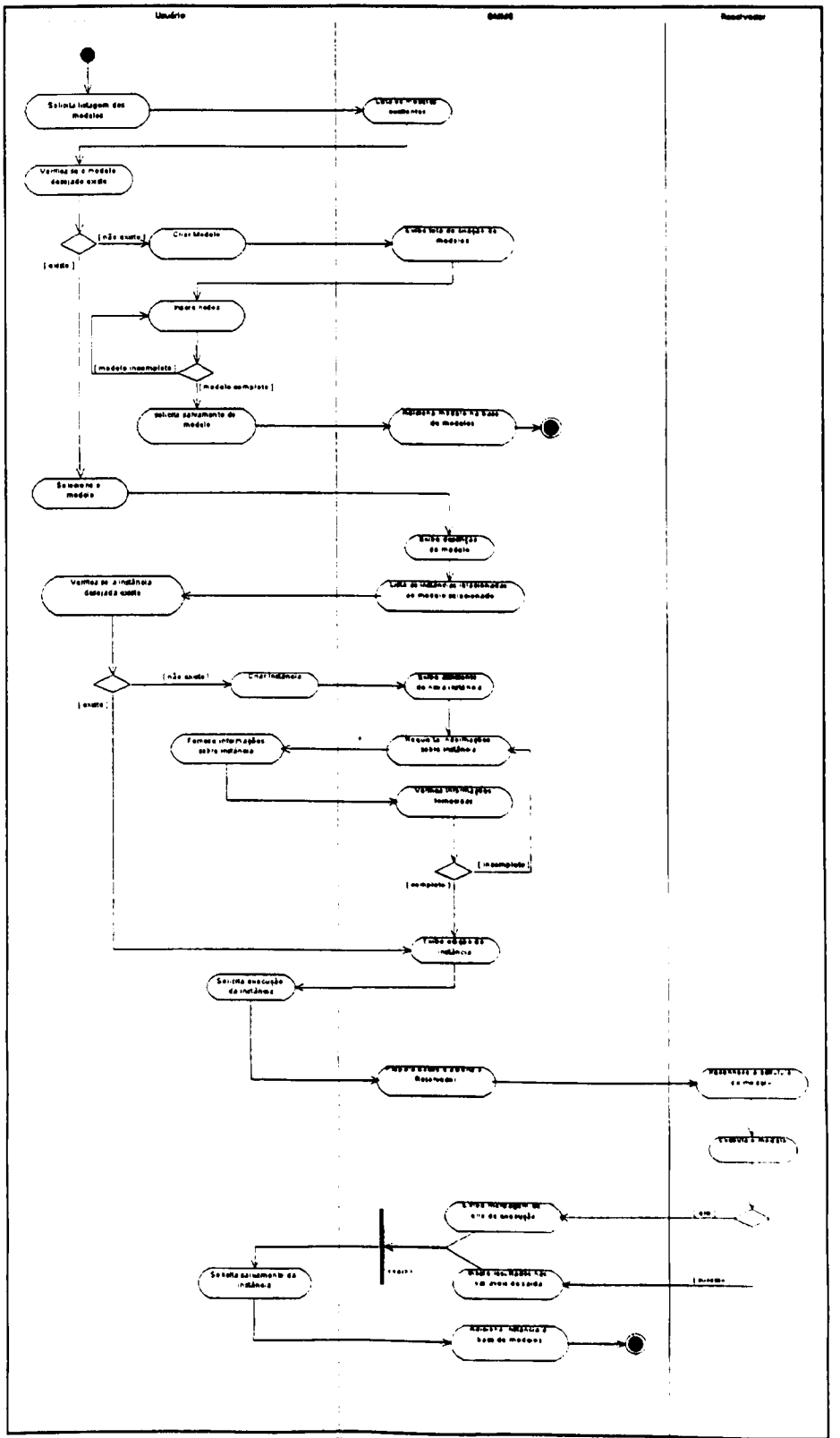


Figura 3.15 – Fluxo de atividades no GMMS

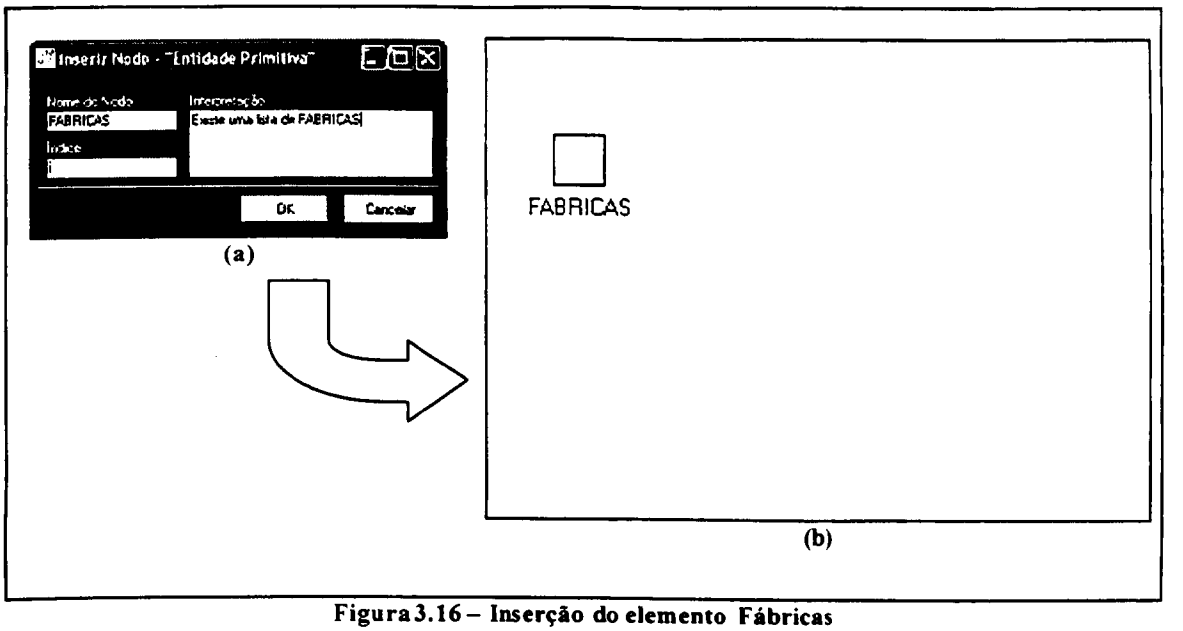


Figura 3.16 – Inserção do elemento Fábricas

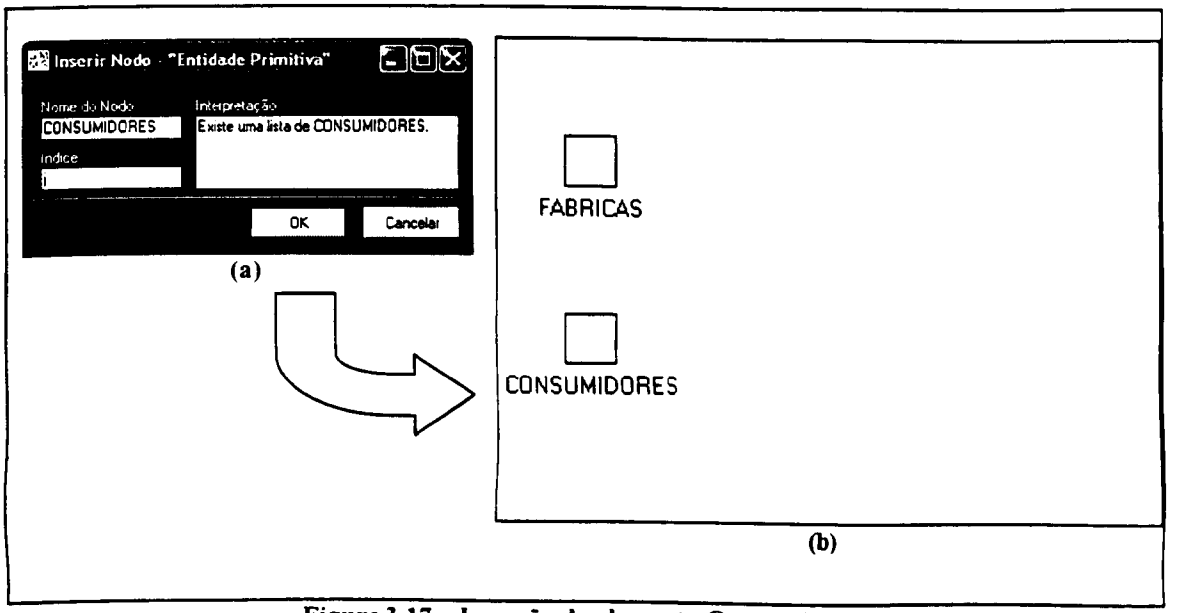


Figura 3.17 – Inserção do elemento Consumidores

Seguindo a construção do modelo, o elemento *Fornecimento*, dependente de *Fábrica* é inserido, como mostra a Figura 3.19a. A capacidade de fornecimento é definida por um número real. Como deve existir um fornecimento para cada fábrica, o índice exportado será "FABRICAi". Selecionando o elemento *Fábrica* na listagem de possíveis nodo origem, obtém-se o resultado mostrado na Figura 3.19b.

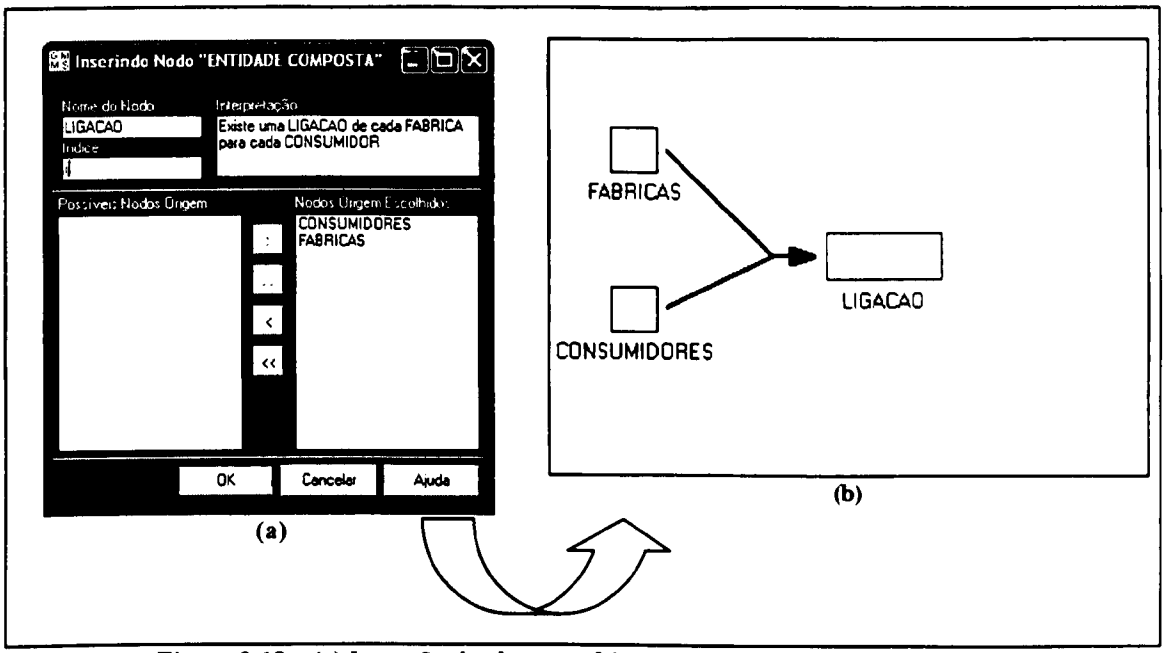


Figura 3.18 – (a) Inserção do elemento Ligações; (b) Resultado no diagrama

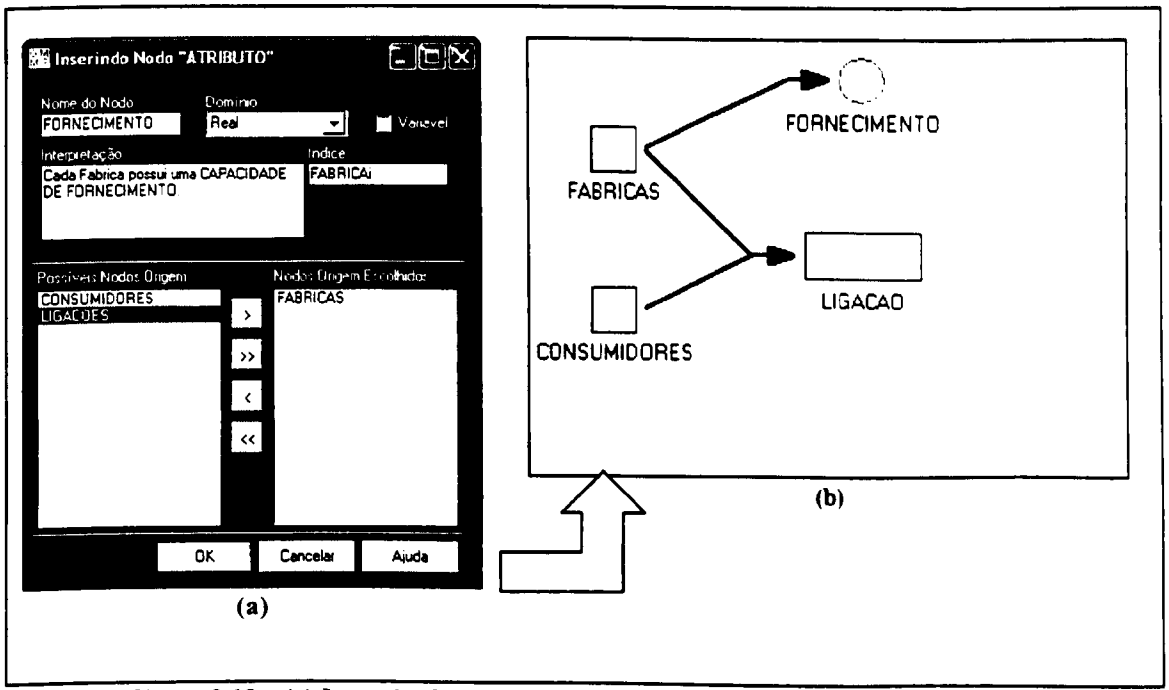


Figura 3.19 – (a) Inserção do elemento Fornecimento; (b) Resultado no diagrama

De maneira análoga a *Fornecimento*, o elemento *Demanda* dependente de *Consumidores* é inserido no diagrama de acordo com a Figura 3.20a, resultando no diagrama da Figura 3.20b. O índice utilizado neste caso, importado de *Consumidores* será "CONSUMIDORESj".

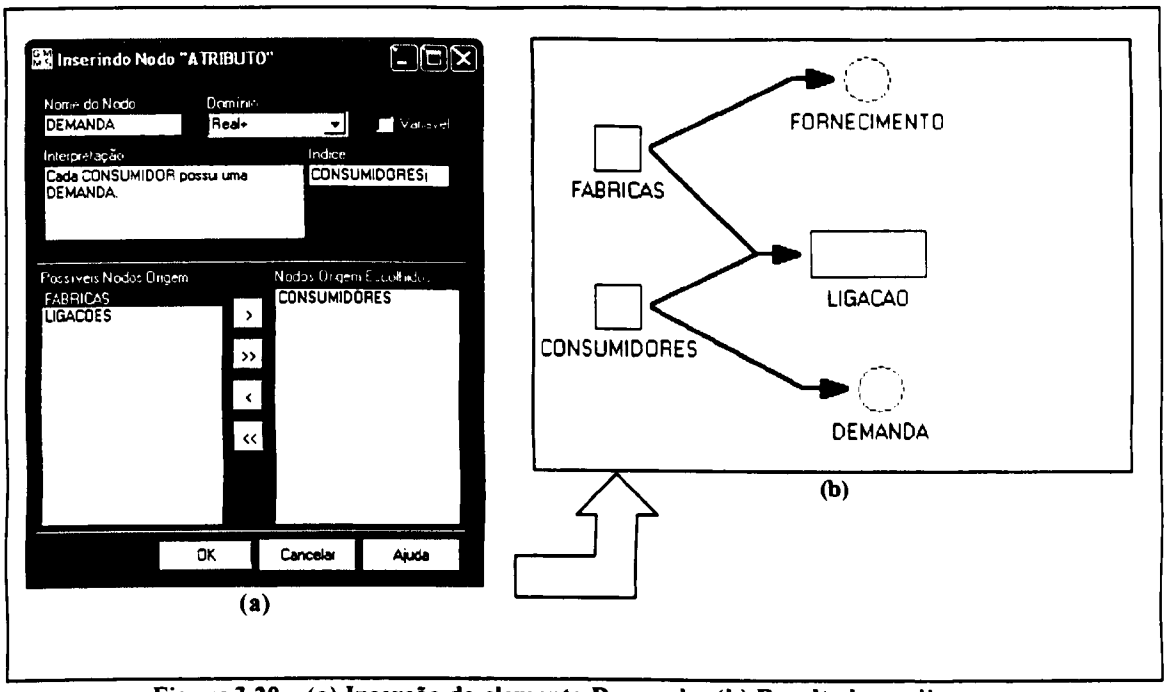


Figura 3.20 – (a) Inserção do elemento Demanda; (b) Resultado no diagrama

Com os elementos relacionados a *Fábrica* e a *Consumidores* já inseridos no diagrama, serão agora inseridos os elementos cuja definição depende de *Ligações*. O primeiro a ser inserido será o elemento *Fluxo*. Como cada elemento de *Ligações* deve possuir um correspondente em *Fluxo*, o índice importado será "LIGACAOij". Nota-se que *Fluxo* está com a caixa de checagem "variável" marcada, o que indica que o tipo deste elemento é atributo variável, ou seja, seus valores estarão sujeitos a constantes alterações (como o caso de simulações e análises *what-if*), ou então seu controle será repassado para o Resolvedor. O processo de inserção de *Fluxo* pode ser acompanhado na Figura 3.21a, o resultado observado na Figura 3.21b.

O segundo elemento dependente de *Ligações* é *Custo*. De maneira análoga à *Fluxo*, *Custo* também possui um elemento correspondente para cada elemento de *Ligações;* em razão disto, seu índice importado também será "LIGACAOij". A Inserção de *Custo* e o resultado no diagrama podem ser observados na Figura 3.22a e Figura 3.22b, respectivamente.

O único elemento do tipo função do modelo, *Custo Total* já pode ser inserido, uma vez que todos os elementos dos quais dependem (*Custo* e *Fluxo*) já estão no diagrama. *Custo Total* é uma função que soma o custo de todos os fluxos, portanto sua expressão, de acordo com a sintaxe da SML, fica "@SUMi SUMj (CUSTOij * FLUXOij)". Como a função retorna um único valor, ao invés de uma lista, seu índice importado será "(CUSTO, FLUXO)". A ausência dos caracteres de índices no final de

cada nome de elemento indica que o *Custo Total* pode ser interpretado como um único valor. A inserção de *Custo Total* pode ser acompanhada na Figura 3.23a, enquanto o diagrama resultante de sua inserção é visto na Figura 3.23b.

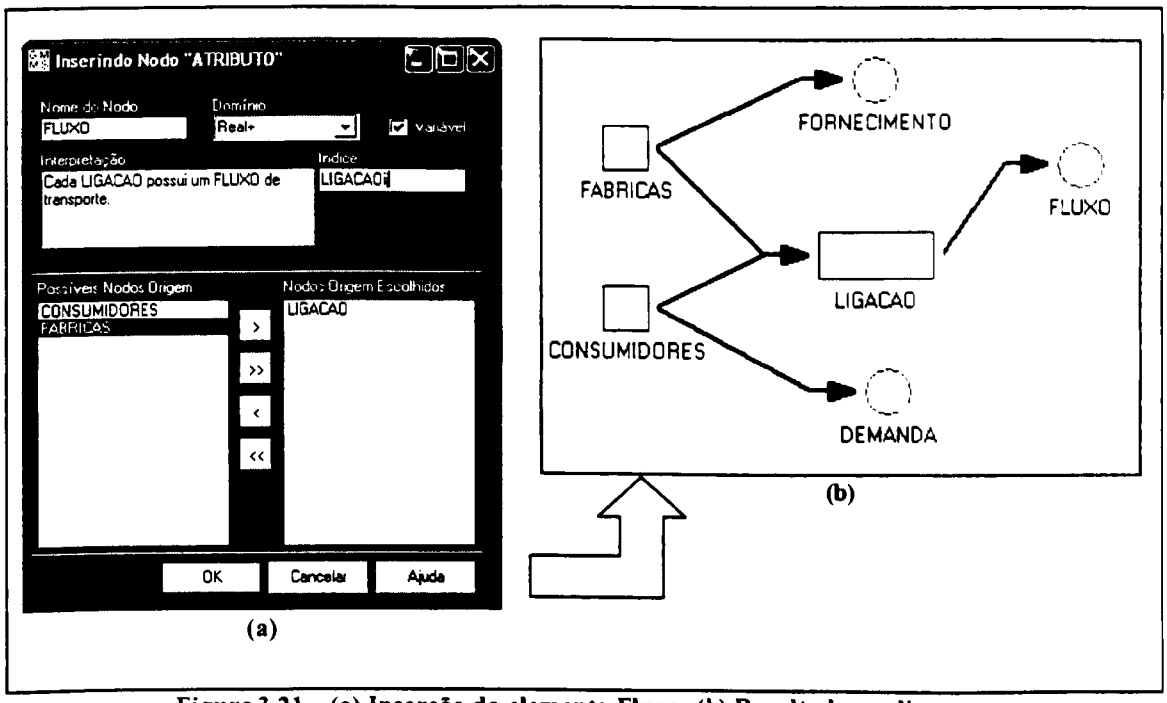


Figura 3.21 – (a) Inserção do elemento Fluxo; (b) Resultado no diagrama

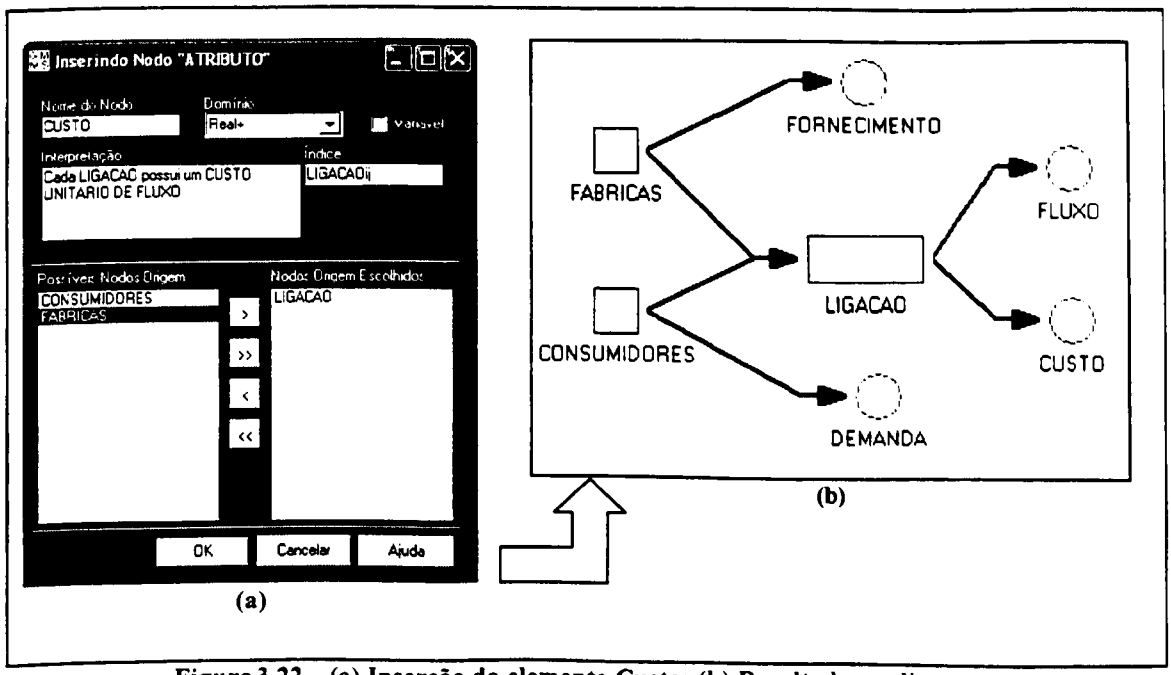


Figura 3.22 – (a) Inserção do elemento Custo; (b) Resultado no diagrama

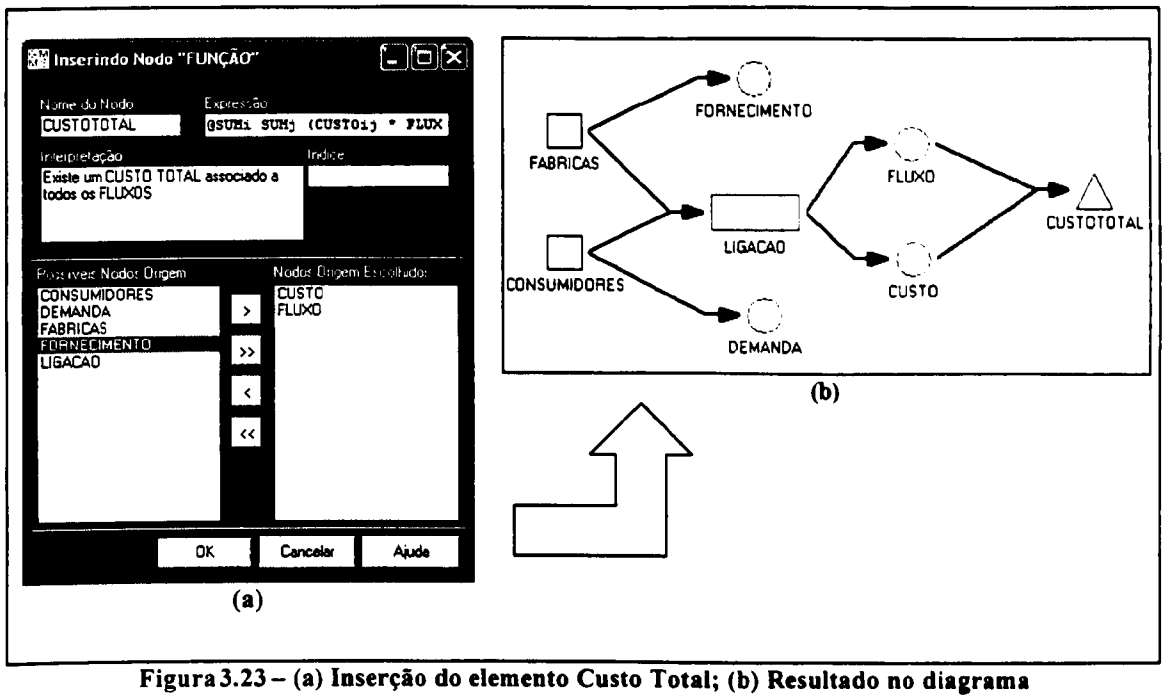


Figura 3.23 – (a) Inserção do elemento Custo Total; (b) Resultado no diagrama

Falta agora inserir as restrições de fornecimento e demanda no modelo. Isto será feito através dos elementos do tipo teste, *Teste de Fornecimento* e *Teste de Demanda*. O primeiro verifica se o fluxo que está deixando uma fábrica é menor ou igual à sua capacidade de fornecimento. O segundo verifica se o fluxo chegando a um consumidor é exatamente igual à sua demanda. O *Teste de Fornecimento* está relacionado a *Fluxo* e *Fornecimento*, deve somar o fluxo total saindo de uma fábrica e compará-lo à capacidade de fornecimento da fábrica, portanto seu índice, ao invés de ser um produto cartesiano como no caso de *Ligações*, será uma projeção de *Fluxo* sobre *Fábricas* (índice "i."), de modo que seu índice importado será "FLUXOi., FORNECIMENTOi", e sua expressão será "@SUMj (FLUXOij) <= FORNECIMENTOi". O acompanhamento da inserção de *Teste de Fornecimento* no GMMS pode ser acompanhado na Figura 3.24a, enquanto o diagrama resultante pode ser visto na Figura 3.24b.

Seguindo o mesmo raciocínio, o *Teste de Demanda* está relacionado à *Fluxo* e *Demanda*. Seu índice importado será "FLUXO.j, DEMANDAj" e sua expressão será "@SUMi (FLUXOij) = DEMANDAj)". O acompanhamento da inserção de *Teste de Demanda* no GMMS pode ser acompanhado na Figura 3.25a, enquanto o diagrama final do modelo, resultante da inserção de seu último componente, pode ser visto na Figura 3.25b.

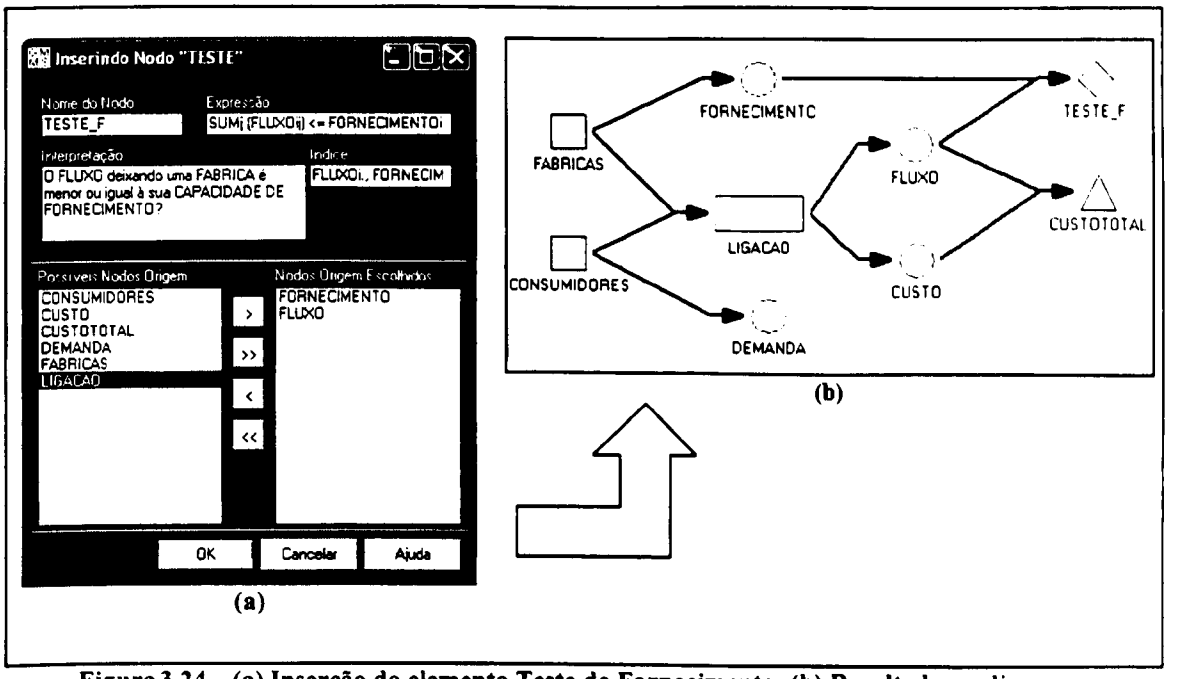


Figura 3.24 – (a) Inserção do elemento Teste de Fornecimento; (b) Resultado no diagrama

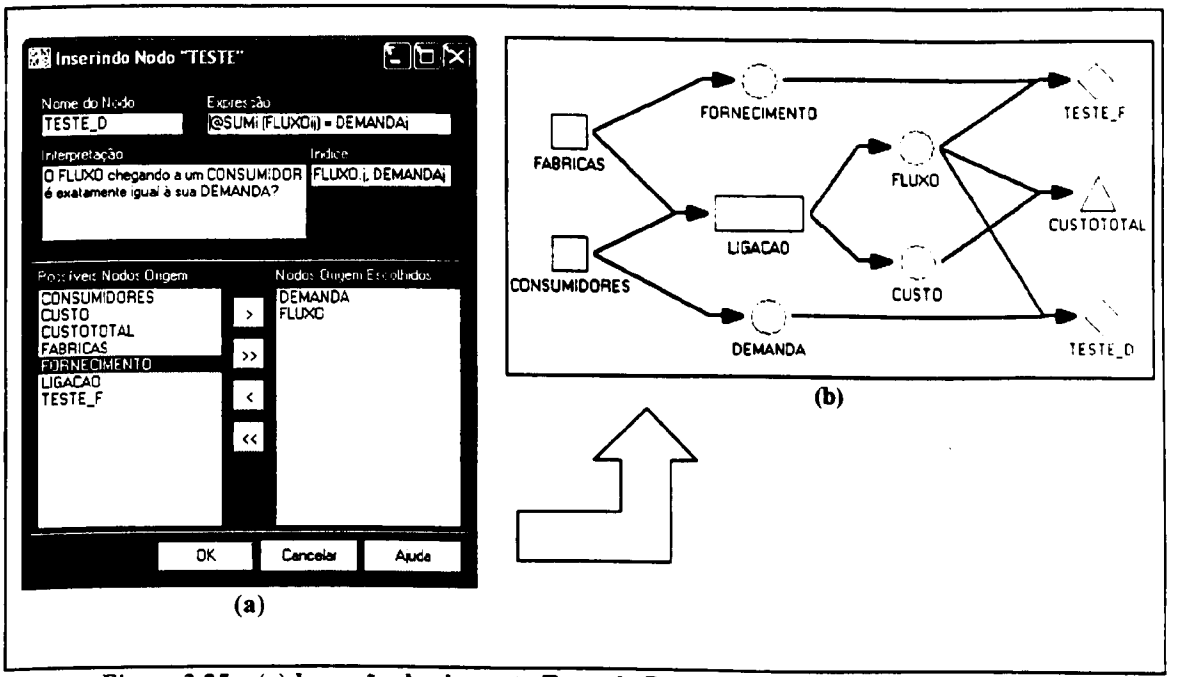


Figura 3.25 – (a) Inserção do elemento Teste de Demanda; (b) Resultado no diagrama

Ao término da formulação, o modelo pode ser adicionado à base de modelos, utilizando o comando "Salvar", localizado na barra de ferramentas.

Após o modelo ter sido adicionado à base de modelos, uma instância pode ser criada. Novamente na tela de listagem, basta selecionar o modelo "TRANSPORTE", que acabou de ser formulado, e executar o comando "Nova".

Ao acionar o comando, o assistente de nova instância é iniciado mostrando sua tela inicial e uma breve descrição (ao lado esquerdo) sobre suas funções. Cada etapa do

assistente é responsável por um tipo de informação. Na primeira etapa, devem ser fornecidas informações específicas sobre a instância do modelo tais como: nome, descrição e o resolvedor que será utilizado na execução. As informações fornecidas na primeira etapa, mostradas na Figura 3.26, equivale a construção da seguinte estrutura em SQL-M:

```
CREATE MODEL SP_RJ_LIN                                    (EXEMPLO 3.1)
        [INTERFACE()
        MODEL ("TRANSPORTE")
        SOLVER ("Transp_Solver")]
```

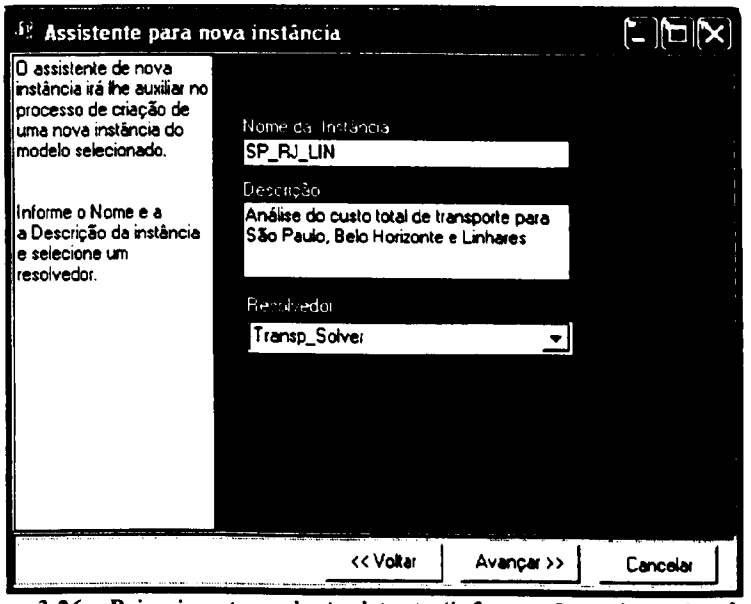


Figura 3.26 – Primeira etapa do Assistente (informações sobre a instância)

Na segunda etapa deve ser indicada a interface da instância. De acordo com as características dos elementos, as variáveis disponíveis para o modelo de transporte são: FORNECIMENTO, DEMANDA, FLUXO, CUSTO, CUSTOTOTAL. Como a única variável de saída é CUSTOTOTAL, basta selecioná-la e direcioná-la para a listagem da direita, como mostra a Figura 3.27. A classificação das variáveis entre entrada e saída é equivalente a construção da cláusula INTERFACE em SQL-M. Ao final desta etapa, a estrutura do exemplo 3.1 será complementada com a cláusula recém construída e, estará completa, como mostra o exemplo 3.2:

Pode-se perceber que a estrutura SQL-M apresentada no exemplo 3.2 corresponde aquela apresentada na seção 2.5.1 para a criação de um modelo em SQL-

M, com a inclusão dos dois nodos do tipo teste que desta vez estão inclusos na interface do modelo como variáveis de saída.

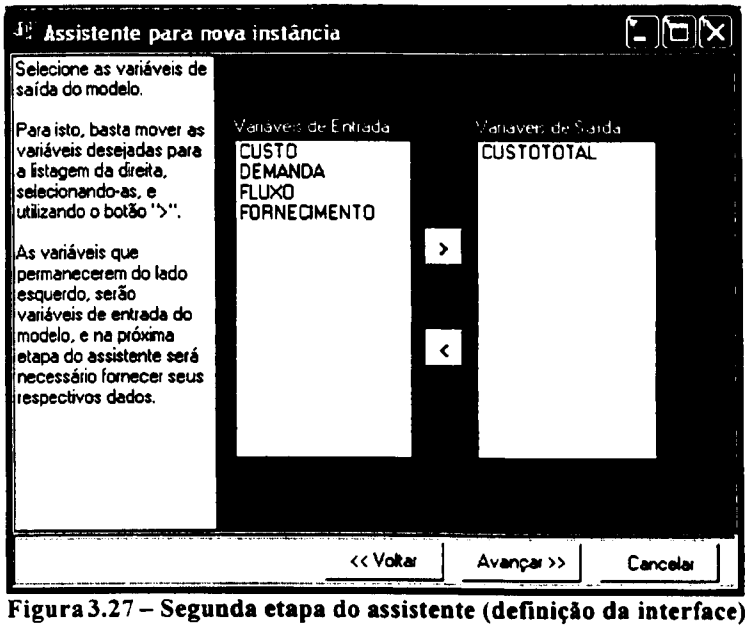


Figura 3.27 – Segunda etapa do assistente (definição da interface)

```
CREATE MODEL SP_RJ_LIN                                    (EXEMPLO 3.2)
    [INTERFACE
        (FORNECIMENTO ARRAY [] OF NUMERIC INPUT,
        DEMANDA ARRAY [] OF NUMERIC INPUT,
        FLUXO ARRAY [][] OF NUMERIC INPUT,
        CUSTO ARRAY [][] OF NUMERIC INPUT,
        CUSTOTOTAL NUMERIC OUTPUT,
        TESTE_F BOOLEAN OUTPUT,
        TESTE_D BOOLEAN OUTPUT)
    MODEL ("TRANSPORTE")
    SOLVER ("Transp_Solver")]
```

Na terceira etapa do assistente, as variáveis devem ser instanciadas. A Figura 3.28 mostra a situação quando falta instanciar apenas a variável CUSTO. Note que sua situação está marcada como "incompleta". Para instanciar a variável CUSTO, será construída uma consulta SQL através da tela de criação de consultas. Supondo uma estrutura de banco de dados semelhante à encontrada na Figura 2.7, a tela de construção visual de consulta SQL teria o aspecto da Figura 3.29 com a consulta pronta. Fechando

a tela de criação de consultas e voltando o controle para o assistente, os dados da consulta podem ser vistos na tabela abaixo das variáveis, como mostra a Figura 3.30. Como agora todas as variáveis foram instanciadas, o assistente pode encerrar seu trabalho e abrir a tela de edição de instâncias. Note que a terceira etapa do assistente corresponde a seguinte construção em SQL-M.

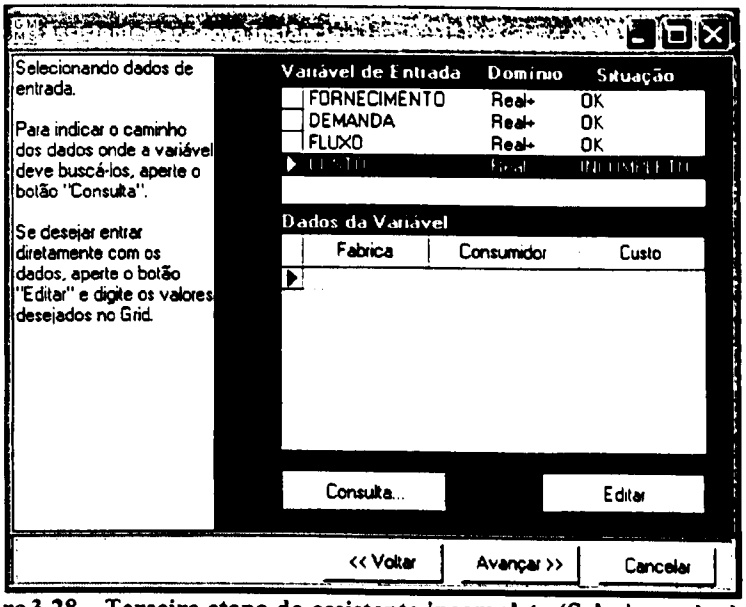


Figura 3.28 – Terceira etapa do assistente incompleta (Selecionando dados)

```
SELECT TOTALCOST, TESTE_D, TESTE_F                    (EXEMPLO 3.3)
FROM TRANSP
SET   FORNECIMENTO = (SELECT Fabrica.FABRICA AS Fabrica.Fabrica,
                      Fabrica.FORNECIMENTO AS Fabrica.Fornecimento
                      FROM ":TRANSPORTE:FABRICA.db" AS Fabrica)
      DEMANDA = (SELECT Consumidores.CONSUMIDORES AS
                      Consumidores.Consumidores,
                      Consumidores.DEMANDA AS Consumidores.Demanda
                      FROM ":TRANSPORTE:CONSUMIDORES.db" AS
                      Consumidores)
      FLUXO = (SELECT Ligacao.FABRICA AS Ligacao.Fabrica,
                      Ligacao.CONSUMIDOR AS Ligacao.Consumidor,
                      Ligacao.FLUXO AS Ligacao.Fluxo
                      FROM ":TRANSPORTE:LIGACAO.db" AS Ligacao)
      CUSTO = (SELECT Ligacao.FABRICA AS Ligacao.Fabrica,
                      Ligacao.CONSUMIDOR AS Ligacao.Consumidor,
```

```
Ligacao.CUSTO AS Ligacao.Custo
FROM ":TRANSPORTE:LIGACAO.db" AS Ligacao))
```

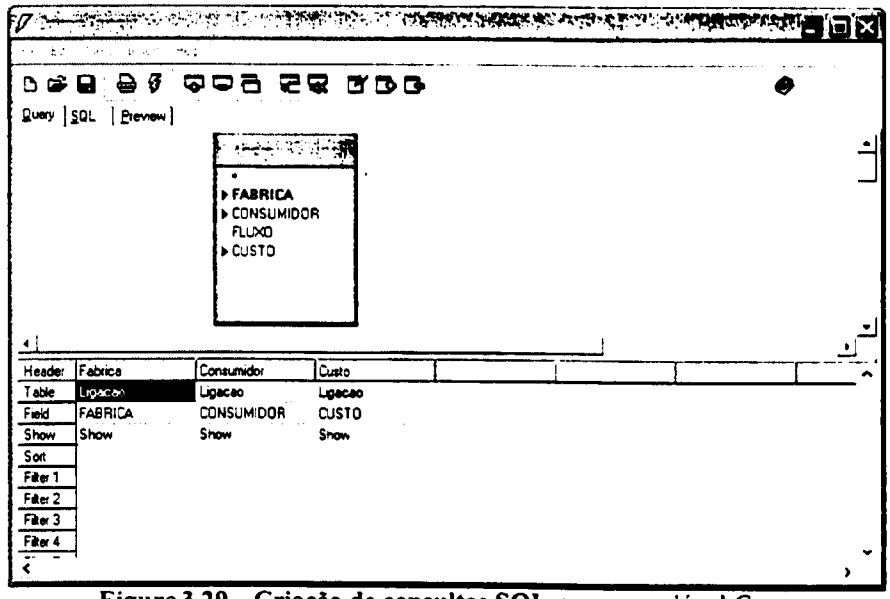


Figura 3.29 – Criação de consultas SQL para a variável *Custo*

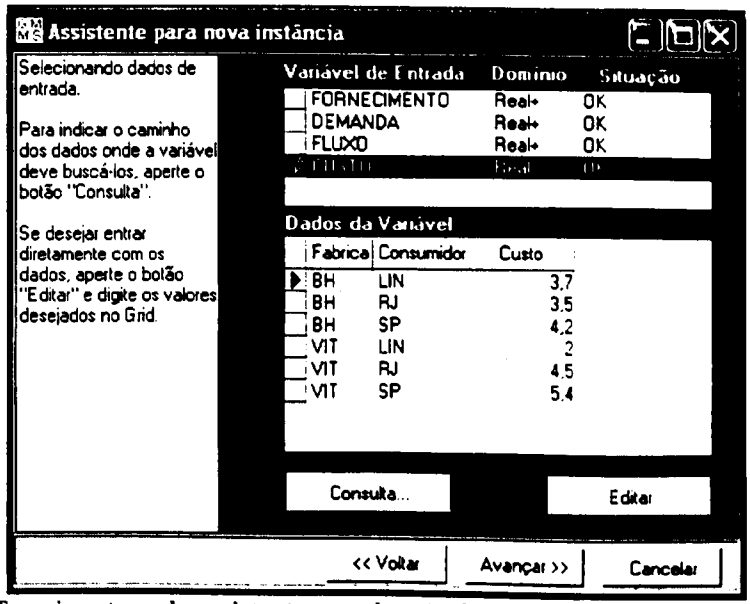


Figura 3.30 – Terceira etapa do assistente completa (todas as variáveis de entrada instanciadas)

Todas as informações fornecidas durante o assistente podem ser alteradas na tela de edição. Pode-se notar que os elementos do tipo teste, apesar de não aparecerem durante o assistente, podem ser visualizados na interface da instância do modelo criado como variáveis de saída. Isto permite ao usuário do GMMS verificar os valores dos testes, o que é de grande utilidade durante simulações e análises *what-if*. Antes da execução, apenas as variáveis de entrada possuem dados para visualização. A tela de

edição de instâncias, antes da execução pode ser vista na Figura 3.31a. Após a execução bem sucedida, as variáveis de saída são instanciadas e também ficam disponíveis para visualização, como na Figura 3.31b.

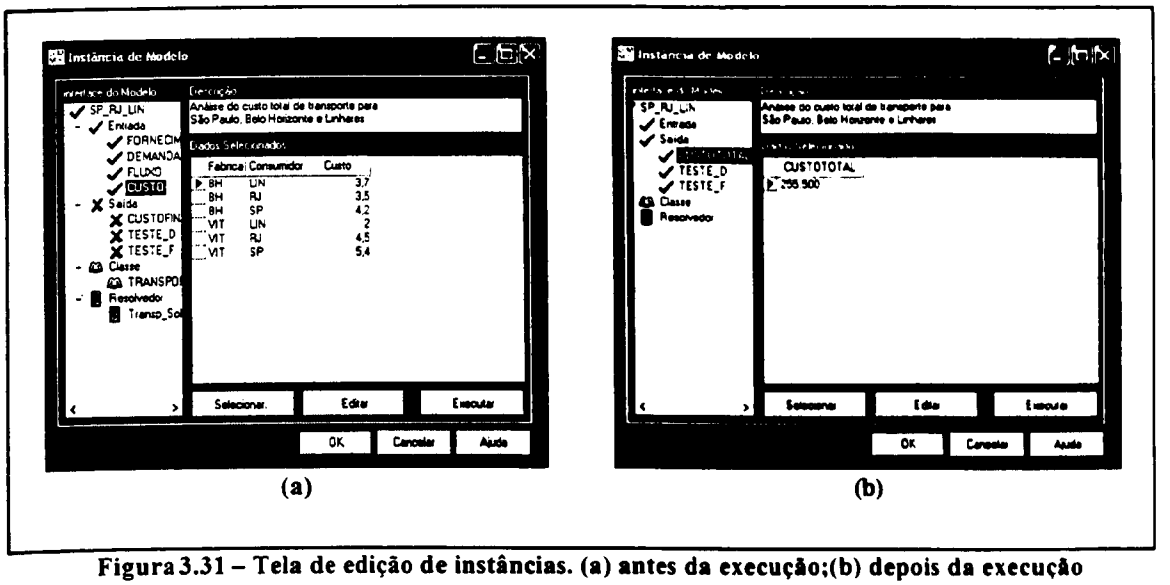


Figura 3.31 – Tela de edição de instâncias. (a) antes da execução;(b) depois da execução

Na tela de edição de instâncias ainda é possível executar simulações e análises *what-if* com a instância, alterando os dados de entrada e executando novamente o modelo. Por exemplo, no modelo de transporte, os valores dos fluxos podem ser alterados, e o resultado será refletido na próxima execução da instância, como mostra a Figura 3.32.

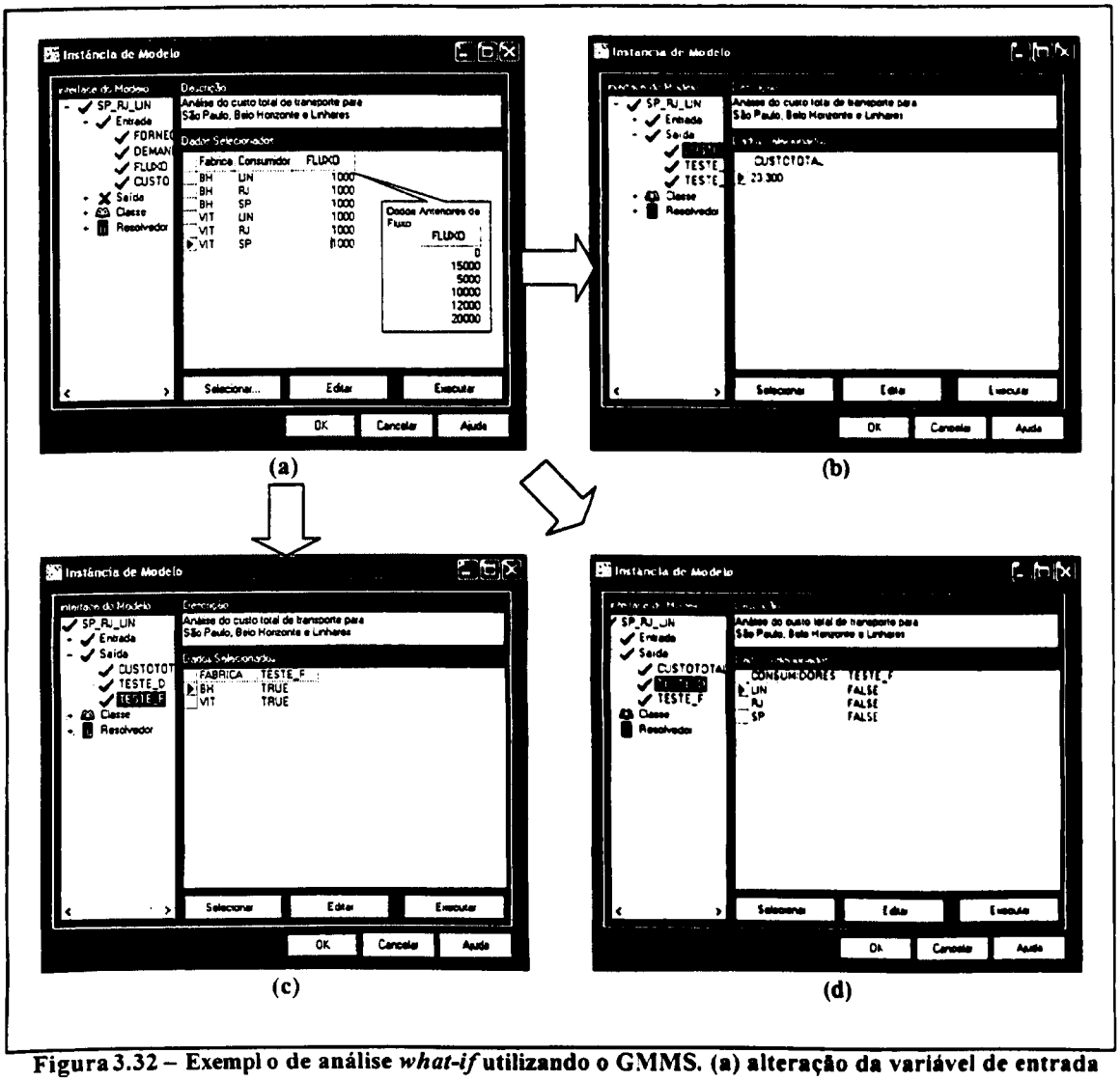


Figura 3.32 – Exemplo de análise *what-if* utilizando o GMMS. (a) alteração da variável de entrada *Fluxo*; Efeito da nova execução nas variáveis de saída (b) *CustoTotal*, (c) *Teste_F*, e (d) *Teste_D*.

4 – Implementação do GMMS

No projeto do GMMS foi utilizada a UML (*Unified Modeling Language*) através de seus diversos diagramas. A escolha da UML se deve ao fato de sua grande aceitação atual nos ambientes de desenvolvimento de softwares, e na facilidade de obtenção de ferramentas que a suportem.

Muitas decisões foram tomadas durante o desenvolvimento do GMMS, que afetaram o seu perfil. Como a princípio foi definido que o GMMS deveria suportar a linguagem SQL-M e complementá-la com uma abordagem gráfica, surgiu a necessidade de definir uma linguagem de definição de modelos para incorporar ao GMMS. Após um prévio estudo das alternativas, foi decidida a utilização da SML devido a seu grande poder de expressão, que permite a representação de modelos de diversas áreas de conhecimento, e crescente utilização em diferentes outros protótipos (mencionados na seção 3.1).

4.1 – Visão dos Casos de Uso

No levantamento de requisitos do GMMS, foi feita uma pesquisa em outros protótipos existentes que utilizam SML (ex. GBMS/SM, IGOR), e na literatura sobre gerência de modelos. Depois desta pesquisa inicial, foram encontrados os casos de uso de um sistema com as características do GMMS, como mostrado na Figura 4.1. Apesar da visão em alto nível, todas as funcionalidades diferentes que o usuário pode obter do GMMS estão representadas.

A descrição de cada um dos elementos do diagrama de casos de uso é mostrada a seguir:

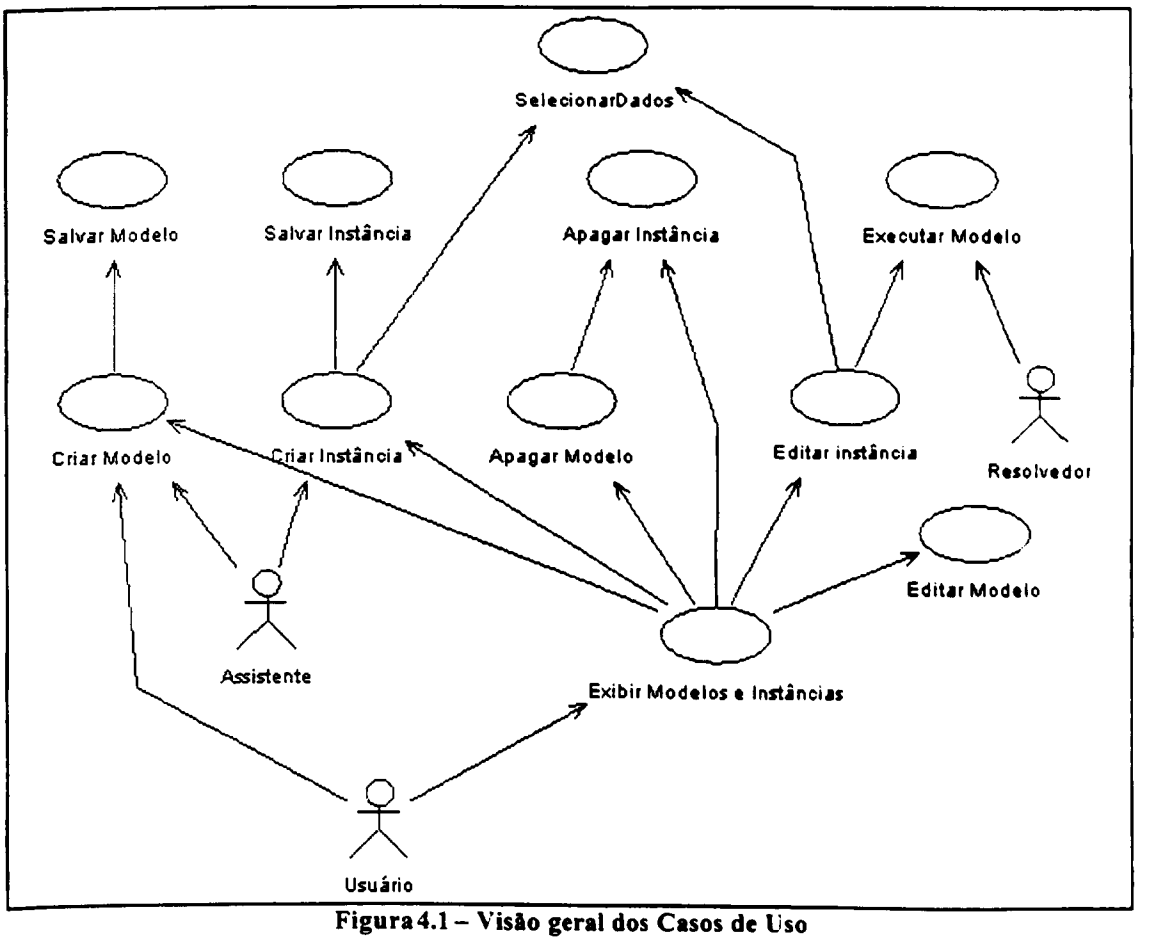


Figura 4.1 – Visão geral dos Casos de Uso

ATORES

- **Usuário** – Usuário do sistema, normalmente será o encarregado pela tomada de decisões dentro das organizações. No sistema, será o responsável pela manutenção e gerenciamento de modelos e instâncias. Deve possuir familiaridade com o ambiente Windows, e algum conhecimento sobre modelagem e bancos de dados.
- **Assistente** – Entidade interna do GMMS responsável pela aplicação do conhecimento sobre SML e sobre SQL-M no GMMS. Atua nas etapas de criação do modelo, realizando a checagem direta de sintaxe, e na criação de instâncias, guiando o usuário através de algumas etapas para a obtenção de uma instância do modelo. Também atua verificando a consistência entre os tipos de dados fornecidos pelo Usuário, e o tipo de dados real das variáveis de entrada do modelo.
- **Resolvedor** – Entidade externa ao sistema, atua na execução de alguma instância de modelo. Deve ler as variáveis de entrada, a estrutura e

fornecer uma resposta que será utilizada para instanciar as variáveis de saída do modelo.

CASOS DE USO

Exibir modelos e Instâncias
Atores Usuário
Pré-Condições Usuário requisita uma exibição da listagem dos modelos cadastrados
Fluxo de Eventos **Principal** 1. Sistema recupera listagem de modelos da base de modelos 2. Sistema recupera listagem de instâncias da base de modelos 3. Sistema verifica o registro de modelo atual e abre uma listagem de Instâncias associadas 4. Sistema exibe listagem de modelos e instâncias **Alternativo** 1. Caso ocorra erro de acesso aos modelos, exibir mensagem de erro.
Pós-Condições Os modelos e instâncias são exibidos na tela.

Criar Modelo
Atores Usuário, Assistente
Pré-Condições Usuário requisita a criação de um novo modelo
Fluxo de Eventos **Principal** 1. Sistema exibe tela de criação de modelos 2. Usuário insere nodo representando elementos da SML 3. Assistente verifica a validade das informações fornecidas pelo Usuário 4. Sistema insere o nodo no diagrama 5. Repetir processo 2, 3, 4 até o modelo estar pronto. **Alternativo** 1. Usuário desiste da criação do modelo e fecha a tela de criação de modelo 2. Sistema volta para a tela principal
Pós-Condições Um esquema de modelo sintaticamente correto

Salvar Modelo
Atores Usuário
Pré-Condições Usuário solicita o salvamento do modelo.
Fluxo de Eventos **Principal** 1. Sistema exibe a tela de informações para salvar o modelo 2. Usuário fornece informações requisitadas 3. Sistema verifica se já existe registro com as mesmas informações 4. Sistema salva o modelo na base de modelos **Alternativo** 3. Caso já exista um modelo com as mesmas informações fornecidas no passo 2, então o sistema exibe mensagem de erro e requisita novas informações.
Pós-Condições Modelo salvo na base de modelos disponível para utilização.

Criar Instância
Atores Usuário, Assistente
Pré-Condições Usuário solicita o criação de uma nova instância
Fluxo de Eventos **Principal** 1. Sistema exibe a tela inicial do assistente de nova instância. 2. Usuário fornece informações requisitadas durante o assistente. 3. Assistente verifica consistência das informações. 4. Sistema exibe tela de edição de instância com os dados coletados pelo assistente. **Alternativo** 1. Caso o Usuário cancele o assistente, o sistema fecha o assistente e exibe a tela anterior. 2. Caso o Assistente detecte alguma inconsistência, uma mensagem de erro deve ser exibida
Pós-Condições Nova instância exibida na tela de instâncias

Salvar Instância
Atores Usuário
Pré-Condições Usuário solicita o salvamento da instância.
Fluxo de Eventos **Principal** 1. Sistema verifica se já existe registro com as mesmas informações. 2. Sistema salva a instância na base de modelos **Alternativo** 1. Caso já exista uma instância com as mesmas informações fornecidas no passo 1, então o sistema exibe mensagem de erro e requisita novas informações.
Pós-Condições Instância salvo na base de modelos disponível para utilização.

Apagar Modelo
Atores Usuário
Pré-Condições Usuário solicita o apagamento do modelo.
Fluxo de Eventos **Principal** 5. Sistema exibe mensagem de advertência e solicita permissão para continuar. 6. Usuário fornece permissão. 7. Sistema verifica se existe Instâncias associadas ao modelo. 8. Sistema aciona caso de uso Apagar Instancia para cada instância associada ao modelo. 9. Sistema apaga o modelo da base de modelos 10. Sistema exibe mensagem de confirmação da operação **Alternativo** 4. Caso o usuário não dê permissão, nada acontece.
Pós-Condições Modelo e instâncias associadas são apagados da base de modelos.

Apagar Instância
Atores Usuário
Pré-Condições Usuário solicita o apagamento de uma instância.
Fluxo de Eventos **Principal** 1. Sistema exibe mensagem de advertência e solicita permissão para continuar. 2. Usuário fornece permissão. 3. Sistema apaga a instância da base de modelos 4. Sistema exibe mensagem de confirmação da operação **Alternativo** 1. Caso o usuário não dê permissão, nada acontece. 2. Caso este caso de uso tenha sido acionado pelo caso de uso Apaga modelo, os passo 1 e 2 são cancelados
Pós-Condições Instância apagada da base de modelos.

Editar instância
Atores Usuário
Pré-Condições Usuário solicita a edição de uma instância.
Fluxo de Eventos **Principal** 1. Sistema recupera os dados da instância selecionada 2. Sistema abre a tela de edição de instância com os dados da instância selecionada. 3. Usuário altera dados da instância **Alternativo** 1. Caso ocorra erro no acesso aos dados, exibir mensagem de erro
Pós-Condições Instancia modificada

Editar Modelo
Atores Usuário
Pré-Condições Usuário solicita a edição de um modelo.
Fluxo de Eventos **Principal** 4. Sistema recupera os dados do modelo selecionado 5. Sistema abre a tela de criação de modelo com os dados do modelo selecionado. 6. Usuário altera dados do modelo **Alternativo** 2. Caso ocorra erro no acesso aos dados, exibir mensagem de erro
Pós-Condições Modelo modificado

Selecionar Dados
Atores Usuário, Assistente
Pré-Condições Usuário utiliza opção de selecionar dados.
Fluxo de Eventos **Principal** 1. Sistema exibe tela de criação de consultas SQL 2. Usuário monta visualmente sua consulta 3. Sistema lê a consulta gerada e recupera os dados selecionados. 4. Assistente verifica se os dados são compatíveis com a variável de saída 5. Sistema utiliza os dados recuperados para instanciar a variável de saída **Alternativo** 1. Caso os dados não sejam compatíveis no passo 4, exibir mensagem de erro.
Pós-Condições Variável de saída instanciada.

Executar Modelo
Atores Usuário, Resolvedor
Pré-Condições Usuário solicita execução do modelo, a partir da tela de edição de instância.
Fluxo de Eventos **Principal** 1. Sistema recupera informações da instância. 2. Sistema recupera informações do modelo associado a instância 3. Sistema organiza as informações em um formato que o Resolvedor possa ler 4. Resolvedor acessa as informações e executa os cálculos necessários 5. Resolvedor fornece um resultado 6. Sistema lê o resultado e utiliza para instanciar as variáveis de saída do modelo 7. Sistema exibe os resultados **Alternativo** 1. Caso aconteça erro de acesso aos dados, exibir mensagem de erro. 2. Caso ocorra um erro durante a execução do resolvedor, exibir mensagem de erro.
Pós-Condições Instância com as variáveis de saída instanciadas
Pós-Condições Instância com as variáveis de saída instanciadas

4.2 – Visão Lógica

As classes no GMMS se dividem em alguns pacotes lógicos, de acordo com sua função. Devido as características do GMMS, estes pacotes foram divididos em: gerência de modelos e interface. O primeiro trabalha diretamente relacionado às funções de gerenciamento da base de modelos, fornecendo mecanismos para o armazenamento e recuperação dos modelos e instâncias criados através do GMMS. Na Figura 4.2, pode-se observar as principais classes da gerência de modelos. Pode-se perceber, pelo diagrama apresentado, que a independência entre modelos, dados e resolvedores foi mantida. Por exemplo, a classe *Instancia*, que representa uma instancia completa de um modelo, utiliza a classe *TEsquemaModelo*, que representa uma classe de modelo, de forma que uma mesma classe de modelo pode ser utilizada em diversas instâncias. Da mesma

maneira, a classe *Instancia* utiliza a classe *Resolvedor*, que representa um resolvedor, de forma que uma instância de modelo possa utilizar diferentes resolvedores.

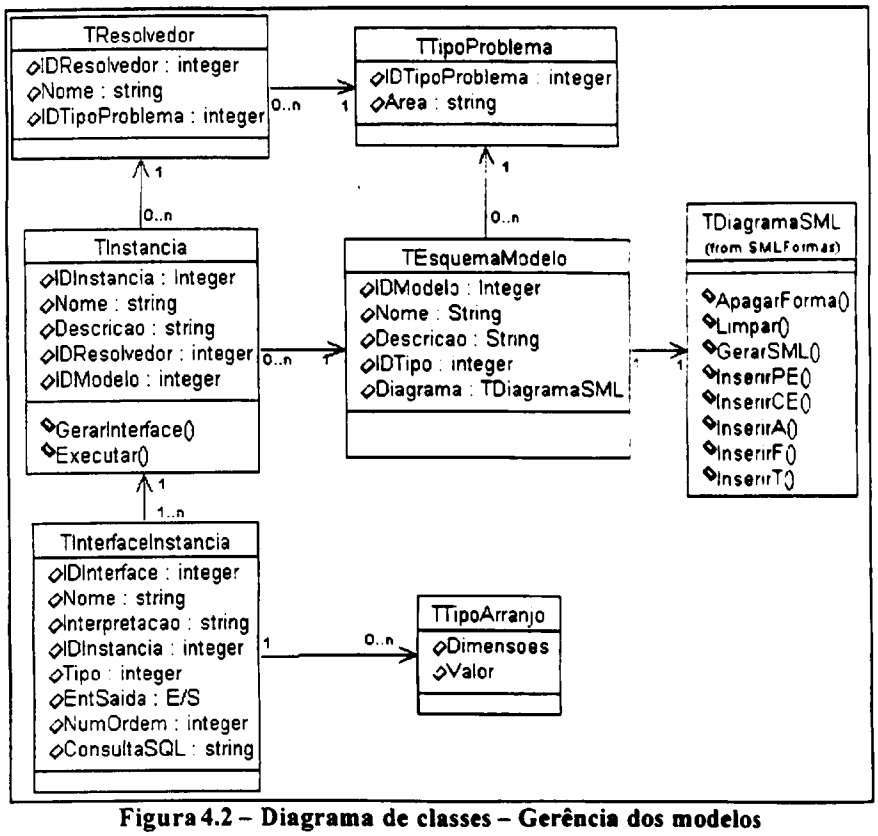


Figura 4.2 – Diagrama de classes – Gerência dos modelos

A classe *TInterfaceInstancia* representa a interface da instância do modelo, ou seja, as variáveis de entrada e de saída. Estas variáveis podem ter uma ou várias dimensões. Para representar as variáveis com mais de uma dimensão, foi incluída a classe *TTipoArranjo*, associada à classe *TInterfaceInstancia*, de maneira que sempre que uma variável de mais de uma dimensão tiver que ser instanciada, a classe *TInterfaceInstancia* automaticamente cria uma instancia da classe *TTipoArranjo* para representá-la. Desta forma, a compatibilidade com a SQL-M se manteve, uma vez que o GMMS também suporta a extensão do subsistema de tipos, como definido por [Pereir97].

Uma representação das classes de gerência de modelos no modelo relacional é quase direta, bastando criar uma tabela para cada uma das classes e utiliza os atributos como colunas destas tabelas. Um dos problemas encontrados foi a representação do atributo *Diagrama,* pertencente à classe *TEsquemaModelo*. A solução encontrada para este problema foi utilizar a classe *TPersistent*,, cuja função é fornecer mecanismos para que seus objetos possam ler e escrever seus atributos publicados (*published*) através de

um fluxo. Portanto, na classe *TEsquemaModelo*, o atributo Diagrama foi representado no modelo relacional através de um nome de arquivo onde as informações dos objetos pertencentes ao diagrama foram salvos. O relacionamento da classe *TPersistent* com a classe *TDiagramaSML* será melhor explicado mais adiante, junto com as classes de interface.

O pacote de interface é utilizado para fornecer suporte às tarefas desempenhadas pelo usuário do GMMS. Uma das principais funções deste pacote é a gerência da interface de criação de modelos, que controla os processos envolvendo a formulação gráfica do modelo. A Figura 4.3 mostra as principais classes envolvidas na interface de criação dos modelos, onde somente os atributos e operações pertinentes às tarefas relacionadas à SML foram mantidos; os demais, que se referem ao controle visual dos componentes, foram suprimidos para melhorar a visualização do diagrama. Vale ressaltar a ligação entre as classes de interface e as classes de gerência de modelos, efetuada através da classe *TDiagramaSML*. Esta classe representa grande parte do suporte a SML fornecido pelo GMMS. Ela é responsável pela gerência dos modelos gráficos, e é através dela que a classe *TEsquemaModelo* pode traduzir a representação gráfica do GMMS para a representação textual, na forma de parágrafos em SML. O trabalho desta tradução foi dividido entre os diversos elementos que compõem o diagrama. Cada nodo representando um parágrafo da SML é o responsável pela criação de sua forma textual através da função *GeraSML* Esta função analisa a estrutura do nodo e produz como saída um parágrafo utilizando a sintaxe da SML. Para a classe *TDiagramaSML* restou o trabalho de organizar a ordem desta tradução, efetuando um caminhamento no diagrama de modo que, considerando o diagrama como uma árvore, um nodo pai sempre seja considerado antes de seus nodos filhos. Deste modo, uma das regras da SML, que impede referências à elementos que ainda não foram definidos, é respeitada.

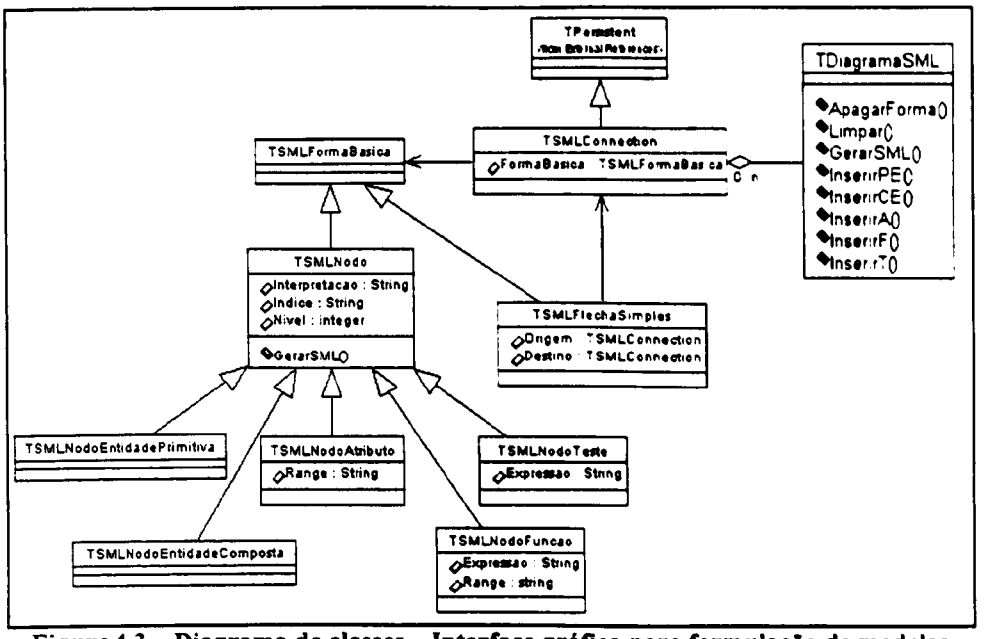


Figura 4.3 – Diagrama de classes – Interface gráfica para formulação de modelos.

A classe *TDiagramaSML* concentra as funções suportadas pelo diagrama, tais como apagar um nodo, limpar a área do diagrama, gerar as cláusulas SML, e as inserções dos diferentes tipos de nodos. Como mostrado no diagrama, ela é uma agregação da classe *TSMLConnection*.

TSMLConnection é descendente da classe *TPersistent*. A função desta classe é prover aos objetos da classe *TSMLFormaBasica*, pontos de conexão que poderão ser utilizados pela classe *TSMLFlechaSimples*. Devido a sua descendência de *TPersistent*, seus atributos podem ser salvos através de um fluxo de dados. Isto é explorado, pela classe *TDiagramaSML*, para salvar todas as informações dos nodos e conexões do diagrama em um arquivo, que fica referenciado pelo atributo *Diagrama* da classe *TEsquemaModelo*.

TSMLFormaBasica é a classe que serve de base para todas as outras formas gráficas do diagrama, como nodos e conectores. Este esquema de herança dos componentes do diagrama de uma única classe ajuda na uniformidade de tratamento dos elementos gráficos, tornando sua implementação e gerenciamento uma tarefa muito mais simples. A função desta classe é fornecer os gerenciamento das características visuais dos nodos (como posicionamento, formato, cores entre outros). As formas visuais que derivam de *TSMLFormaBasica* são :

- **TSMLNodo** – Classe pai dos diferentes tipos de nodos do diagrama. Concentra todo o controle sobre o comportamento gráfico dos nodos, como exibição e movimento. Também possui os atributos comuns aos

elementos da SML, como *Índice* e *Interpretação*. O atributo *Nível* é o responsável pelo controle da hierarquia do modelo. A atribuição de seu valor é feito comparando o nível de um nodo origem, acrescido de um, com o nível do nodo destino, de modo que o maior dos dois valores será mantido como o nível do nodo destino. As classes derivadas de *TSMLNodo* são os diferentes tipos de elementos da SML, representados por:

- **TSMLNodoEntidadePrimitiva** – Representa um elemento do tipo entidade primitiva da SML, seus atributos principais são derivados de *TSMLNodo*. Sua função GeraSML retorna um parágrafo do tipo "/pe/" na sintaxe da SML.
- **TSMLNodoEntidadeComposta** – Representa um elemento do tipo entidade composta da SML, seus atributos principais são derivados de *TSMLNodo*. Sua função GeraSML retorna um parágrafo do tipo "/ce/" na sintaxe da SML.
- **TSMLNodoAtributo** – Representa um elemento do tipo atributo da SML, seus atributos principais são derivados de *TSMLNodo* e mais dois novos são incluídos. O primeiro, chamado *Range*, representa o domínio dos valores que o atributo pode conter. O segundo, chamado *IsVariavel*, indica se o nodo é do tipo atributo variável ou não. Sua função GeraSML retorna um parágrafo do tipo "/a/", ou do tipo "/va/ para o caso de IsVariavel possuir como valor falso, ou verdadeiro respectivamente.
- **TSMLNodoFunção** – Representa um elemento do tipo função da SML, seus atributos principais são derivados de *TSMLNodo*, e da mesma maneira que o nodo atributo, possui o atributo *Range* para especificar o domínio de seus valores. Além de *Range* um outro atributo foi adicionado chamado *Expressao*, que é utilizado para armazenar a expressão de função associada ao nodo. A função de expressão deve adotar a sintaxe da SML, como mostrado em [Geoffr89]. Sua função GeraSML retorna um parágrafo do tipo "/f/" na sintaxe da SML.
- **TSMLNodoTeste** - Representa um elemento do tipo teste da SML, seus atributos principais são derivados de *TSMLNodo*, e da

mesma maneira que o nodo atributo, possui o atributo *Range* para especificar o domínio de seus valores, e do mesmo modo que o nodo função, possui um atributo *Expressao*, que neste caso representa uma expressão de teste. O atributo range presente nos nodos atributo e função não se aplica neste caso, pois o nodo teste já possuiu, por definição, valores lógicos como domínio. Do mesmo modo que a expressão de função, a expressão de teste também deve ser escrita na sintaxe da SML, como mostrado também em [Geoffr89]. Sua função GeraSML retorna um parágrafo do tipo "/t/" na sintaxe da SML.

- **TSMLFlechaSimples** – são os conectores do diagrama. Controlam a exibição dos conectores e seu comportamento, como o seu redesenho quando algum nodo conectado é movimentado. Dentro de sua estrutura possui dois objetos da classe *TSMLConnection*, que servem para especificar a forma de origem e a forma de destino da conexão.

A interação entre o usuário e o GMMS, durante a formulação visual do modelo, pode ser melhor observada através do diagrama de atividade mostrado na Figura 4.4, que descreve o processo de inserção de um nodo no diagrama. A interação entre classes, durante a inserção de um nodo do tipo "pe", pode ser acompanhada na Figura 4.5.

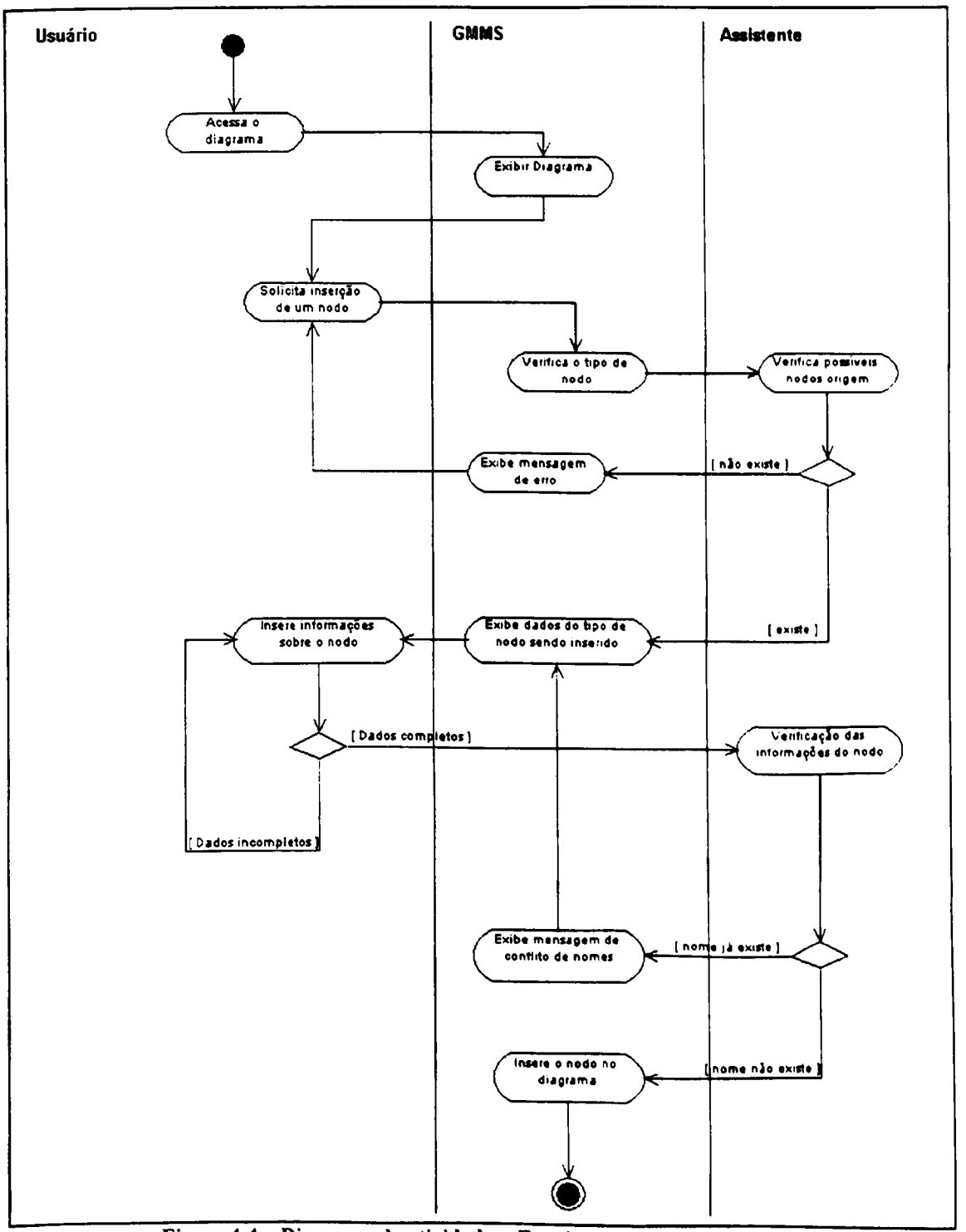


Figura 4.4 – Diagrama de atividade – Funcionamento do diagrama.

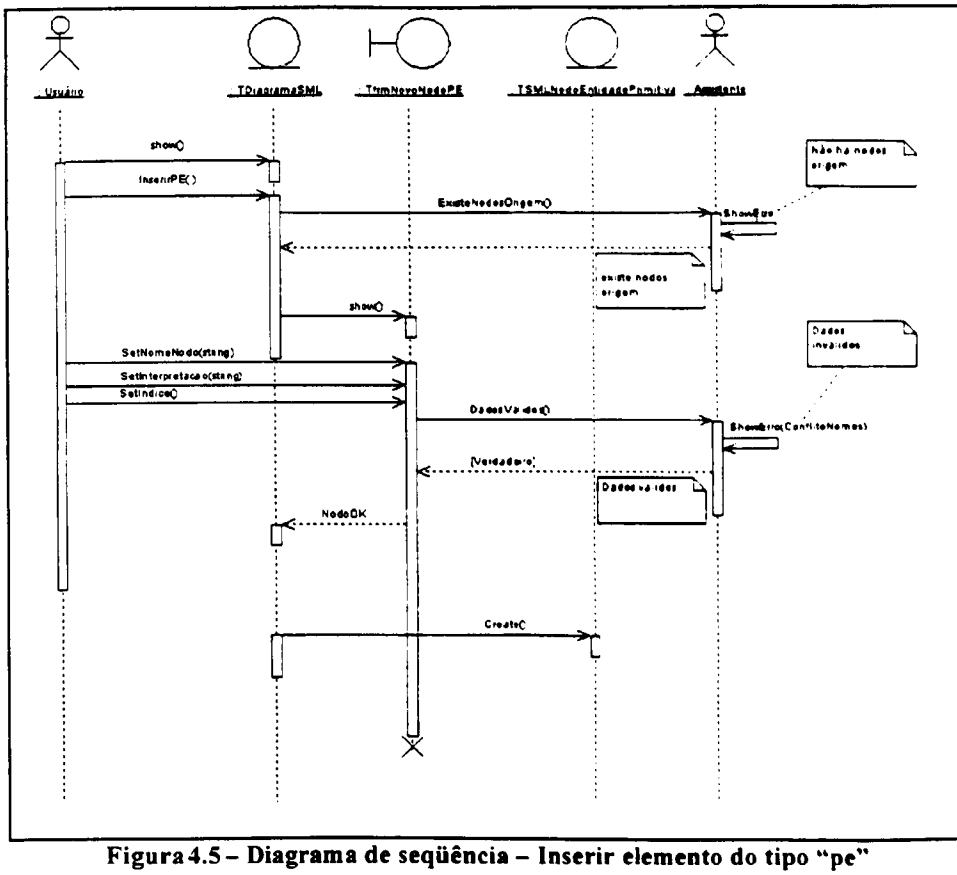


Figura 4.5 – Diagrama de seqüência – Inserir elemento do tipo "pe"

4.3 – Visão dos Componentes

Uma visão dos componentes utilizados na implementação do GMMS pode ser vista na Figura 4.6. Cada elemento do diagrama com o estereótipo "unit" representa um arquivo com código-fonte. O componente com o estereótipo "program" é o módulo principal do GMMS, responsável pela execução da aplicação.

O componente *Power Query* é um componente desenvolvido pela Shazamware, e é responsável pela criação de consultas visuais em SQL disponível no GMMS. Esta ferramenta foi escolhida após uma comparação com diversas alternativas, onde se demonstrou ser a mais completa e intuitiva, e por sua semelhança de funcionamento com o esquema adotado, para criação de consultas, no SQL Microsoft Access, que é uma ferramenta de utilização bastante popular, tendo em vista que faz parte do pacote do Microsoft Office, que é o atual líder do mercado no segmento de software para criação de documentos.

O componente *Resolvedores* são aplicativos fora do ambiente do GMMS, mas que possuem uma interface de forma que seja possível a troca de informações entre um e outro.

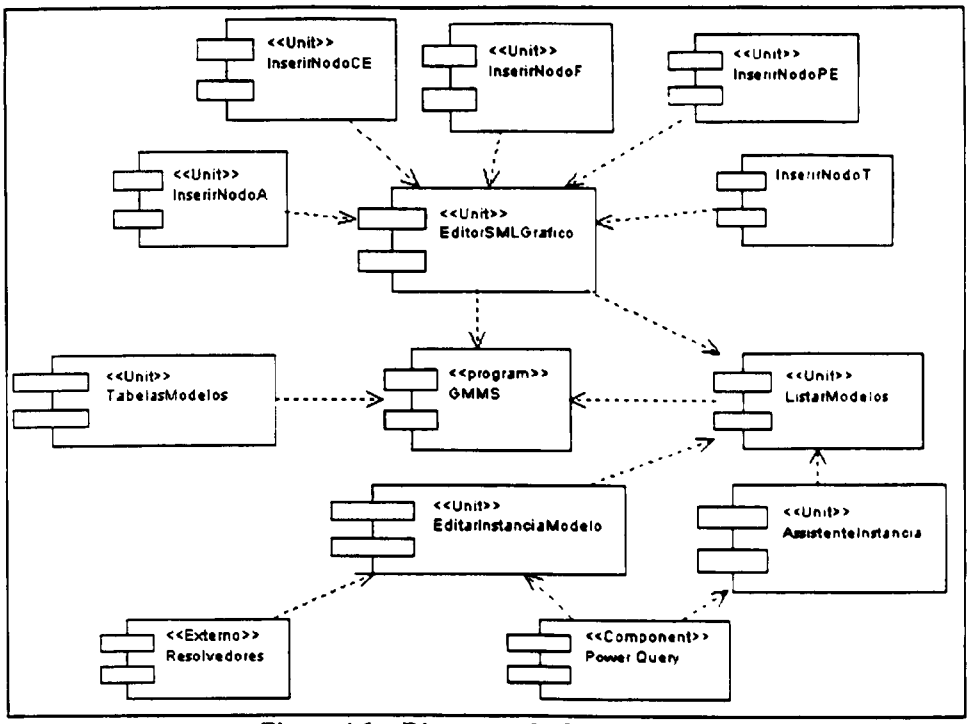


Figura 4.6 – Diagrama de Componentes

4.4 – Processo Adotado

A implementação do GMMS foi realizada de acordo com o processo *Rational Unified Process* (RUP). O desenvolvimento no RUP é realizado através de ciclos, onde cada ciclo resulta em uma nova versão do produto. O ciclo é composto de quatro fases, que são :percepção, elaboração, construção e transição.

Na fase de percepção, foi feito o vislumbre inicial do sistema, sobre como deveria trabalhar e o que deveria fazer. Os principais atores foram selecionados, e em seguida, na fase de elaboração, foi feito o levantamento dos casos de uso, constituindo a visão de casos de uso do sistema. Os casos de uso foram descritos através de diagramas de seqüência, o que facilitou a criação da visão lógica do sistema, que foi representada através de diagramas de classes, e diagramas de atividade. Com o GMMS especificado, iniciou-se a próxima fase.

Na fase de construção, um protótipo da interface do GMMS foi construída, e através de vários ciclos de análise-projeto-programação, o GMMS foi construído de forma incremental e interativa.

Como o objetivo final da implementação do GMMS era a obtenção de um protótipo, a fase de transição foi adaptada para as necessidades do GMMS, onde ao

invés de colocar o GMMS em uso em um ambiente real, foram realizadas outras tarefas incluídas nesta fase, como: definição da instalação e configuração.

4.5 – Considerações Finais

Mesmo utilizando um processo de desenvolvimento, a implementação do GMMS sofreu alguns contratempos imprevistos. A implementação da criação gráfica de modelos levou um tempo maior do que o esperado em relação ao restante do projeto, devido as dificuldades inerentes de se implementar um sistema de diagramação. Como não foi encontrado nenhum componente de terceiros que pudesse ser utilizado para economizar tempo de programação, esta parte foi praticamente toda implementada do zero, consumindo tempo precioso.

Outro problema difícil de ser contornado foi a dificuldade de encontrar informações formais sobre a SML. Apesar de existirem vários materiais disponíveis sobre o assunto, dificilmente algum mostrava mais do que detalhes superficiais. O problema só foi resolvido muito tempo depois que a fase de implementação do GMMS já havia iniciado. Somente com este material em mãos foi possível desenvolver grande parte deste trabalho.

Atualmente, o GMMS conta apenas com resolvedores específicos para determinados modelos (como o modelo de transporte), o que não impede que novos resolvedores de uso mais geral sejam adicionados, bastando para isto prover a comunicação entre o resolvedor e o GMMS.

Mesmo assim, o resultado obtido mostra que a proposta de um ambiente gráfico para gerência de modelos é útil para os tomadores de decisão de uma empresa, não importando a área de atuação.

A arquitetura aberta do GMMS, que possibilita a inclusão de novos modelos, sugere a personalização do ambiente de acordo com as necessidades da organização, ou seja, se a organização necessita de modelos estatísticos, a base de modelos do GMMS pode ser carregada com modelo estatísticos, prontos para a instanciação e execução por resolvedores que os suportem, que também poderiam fazer parte do "pacote" de software do GMMS.

5 – Conclusões

Fornecendo um único ambiente onde os modelos podem ser definidos, instanciados e executados, o ambiente gráfico proposto pode auxiliar os usuários de Sistemas de Apoio a Decisão nas tarefas de formulação e manipulação dos dados de maneira simples e visual. A ambiente gráfico apresentado mostrou-se uma alternativa viável para grande parte dos objetivos desejados em um Sistema de Gerência de Modelos, fornecendo recursos para facilitar a manipulação d modelos e dados, integração com banco de dados, apoio nas tarefas essenciais como formulação e instanciação dos modelos, independência entre modelos, dados e resolvedores, e uma interface gráfica que permite sua utilização por usuários não especialistas.

A manipulação dos modelos é feita de forma visual através de diagramas, representando grafos de modelo, onde auxílio na formulação do modelo é prestado na forma de um analisador direto de sintaxe, que evita que modelos sintaticamente incorretos sejam criados. O suporte à SML oferece um grande poder de expressão, permitindo que modelos das mais diferentes áreas possam ser criados, utilizando uma única linguagem de definição.

A integração da SML com a SQL-M permite a construção de instâncias de modelos de maneira simples, pois sua interface é reconhecida automaticamente pelo ambiente. A integração entre os modelos e os dados, que é um dos problemas encontrados em sistemas de gerência de modelos, foi resolvida através do uso de SQL na instanciação das variáveis de entrada do modelo.

A independência entre modelos, dados e resolvedores pode ser observada na separação que há entre os esquemas de modelos e suas instâncias. Um esquema de

modelo criado pode possuir diferentes instâncias, enquanto cada uma das instâncias de um modelo pode ser resolvida por diferentes resolvedores, desde que os resolvedores sejam adequados para o problema em questão.

A interface gráfica, implementada através do protótipo GMMS, constitui uma importante ferramenta para que usuários não especialistas possam utilizar sistemas de gerência de modelos, de forma segura e eficiente. Os modelos são criados através de diagramas representando grafos de modelo, as instâncias são criadas através de um assistente que auxilia na definição da interface entre a instância, o resolvedor e os dados que serão utilizados. Os dados são selecionados através de um consulta em SQL criada de maneira visual, no mesmo estilo de ferramentas QBE (*Query By Example*). Os dados retornados pelo Resolvedor são utilizados para instanciar as variáveis de saída do modelo, permitindo sua fácil visualização. Devido às facilidades da interface, análises *what-if* podem ser realizadas facilmente para uma instância de modelo.

O objetivo final de um ambiente gráfico para gerência de modelos para utilização de usuários não especialistas foi validado através do protótipo GMMS. Por se tratar de um protótipo, algumas funcionalidades não foram implementadas por razões diversas, mas o resultado final pode ser observado através da interface e de simulações realizadas pelo GMMS. Com mais algum esforço de programação, o GMMS totalmente implementado já seria uma ferramenta de grande ajuda na tomada de decisão das organizações, mas diversas melhorias e sugestões podem ser propostas para a continuidade deste trabalho.

Um sugestão seria a implementação de um protocolo de comunicação entre os resolvedores e os modelos, de forma que pudessem trocar informações, de maneira uniforme, entre os diversos tipos de resolvedores e modelos. Na solução deste problema, poderia ser utilizado um protocolo utilizando a linguagem XML (*eXtended Markup Language*), que tem como uma de suas principais finalidades a estruturação e organização de dados.

Sistemas especialistas podem ser utilizados para auxiliar na escolha de um modelo que resolva um determinado problema. Com este mesmo objetivo, pode-se aplicar ontologias sobre as informações dos modelos para conseguir uma busca mais eficiente. Para viabilizar a utilização de ontologias, seria necessário uma taxonomia dos diferentes tipos de modelo que podem ser encontrados.

Outra sugestão seria a construção de um banco de Resolvedores, para gerenciar os possíveis resolvedores que poderiam ser utilizados para a execução dos modelos.

Mais uma vez, poderiam ser utilizados Sistemas Especialistas e ontologias no processo de escolha de um resolvedor para um determinado problema.

De acordo com a sugestão do final da seção 4.5, o GMMS poderia trabalhar com o esquema de pacotes, de forma que o usuário escolheria que pacotes gostaria de instalar como, por exemplo, pacotes estatísticos, pacotes de otimização, etc. Esta opção abre um enorme campo de atuação para o GMMS que, neste formato, poderia ocupar um papel vital na tomada de decisão das organizações.

6 – BIBLIOGRAFIA

[Bissch94] Bisschop J, Entriken R."AIMMS:The Modeling System". *Haarlem*, The Netherlands: Paragon Decision Technology, 1994.

[Blanni85] Blanning, R.W., "A Relational Framework for Join Implementation in Model Management Systems", *Decision Support Systems*, Vol.1, No.1, janeiro, p69-81, 1985.

[Blanni86a] Blanning, R.W. "A Relational Framework for Information Management", *Decision Support Systems: A Decade in Perspective*, North Holland Publishers, Armsterdam, p25-40, 1986

[Blanni86b] Blanning, R.W.,"An Entity-Relationship Approach to Model Management", *Decision Support Systems*, Vol.2, No.1, março, p64-72, 1986.

[Blanni93] Blanning, R.W., King, D.R. "Introduction: The Growth of Decision Support Technology", *Current research in decision support techology*, IEEE Computer Society Press, p1-8, 1993.

[Brooke88] Brooke, A., D. Kendrick, and A. Meeraus, "GAMS: A User's Guide", *The Scientific Press*, Redwood City, CA, 1988.

[Chari97a] Chari, K., Sen, T. K., "An Integrated Modeling System for Structured Modeling Using Model Graphs", *INFORMS Journal on Computing*, Volume 9, No. 4, 1997.

[Chari97b] Chari, K., Sen, T. K., "A Graphical Modeling System : Applications in Organizations Model Management", *OMEGA*, Volume 25, No. 2, p241-253, 1997

[Chari98] Chari, K., Sen, T. K., "An Implementation of a Graph-Based Modeling System for Structured Modeling (GBMS/SM)", *Decision Support Systems*, Volume 22, Issue 2, Fev 1998, p103-120

[Chen76] CHEN, P, "The Entity Relationship Model: Toward a Unified View of Data", *ACM Transactions on Database Systems*, Vol. 1, No. 1, pp. 9-36, 1976.

[Collau94] Collaud G., Pasquier-Boltuck, J."gLPS: A Graphical Tool for the Definition and Manipulation of Linear Problems." *European Journal of Operational Research*, 72(2):277-86, 1994.

[Dennis88] Dennis, A., et al. "Information Technology to Support Eletronic Meetings." *MIS Quarterly*, Dezembro, 1988.

[Dolk88] Dolk, D. R., "Model management and Structured Modeling: the Role of an Information Resource Dictionary Systems", *IEEE Computer* 17, 9 (September 1984), 89-97.

[Elliso82] Ellison E, Mitra G. "UIMP: User Interface for Mathematical Programming." *ACM Transactions on Mathematical Software*,8:229-55, 1982.

[Elmasr94] Elmasri, R., and S. B. Navathe, "Fundamentals of Database Systems", Benjamim/Cummings, Redwood, CA, 1994.

[Finholt90] Finholt, T., e Sproull, L.S. "Eletronic Groups at Work" *Organization Science*, Vol 1, N 1, 1990.

[Fourer90] Fourer, R., D. M. Gay, and B. W. Kerninghan, "A Modeling Language for Mathematical Programming", *Management Science* 36, 5 (May 1990), 519-554.

[Gagliar96] Gagliardi, M., Spera, C., "MODASS: A Modeling Systems to Manage Structured Models", *Proceedings of the first INFORMS Conference on Information Systems and Thecnology*, May 5-8, Washington DC., p80-89, 1996.

[Gagliar97] Gagliardi, M., Spera, C., "BLOOMS: A Prototype Modeling Language with Object Oriented Features", *Decision Suppport Systems*, n.19, p1-21, 1997.

[Gallup88] Gallupe, B., e DeSanctis, G. "Group Decision Support Systems" *DSS-86 Transactions*, Washington, DC: The Institute of Management Sciences, 1988.

[Geoffr87] Geoffrion, A.M., "An Introduction to Structured Modeling", *Management Science*, Vol. 33, No. 5, Maio, p547-588, 1987.

[Geoffr89] Geoffrion, A.M."The Formal Aspects of Structured Modeling", *Operations Research*, 41, 33-43, 1989.

[Geoffr90] Geoffrion, A.M., "SML: A Model Definition Language for Structured Modeling",Working Paper 360, Western Management Science Institute, UCLA, last revision August, 1990.

[Geoffr91] Geoffrion, A.M."FW/SM: A Prototype Structured Modeling Environment", *Management Science*, 37, 1513-1538, 1991.

[Geoffr92a] Geoffrion, A.M."The SML Language for Structured Modeling: Levels 1 and 2", *Operations Research*, 40:1, janeiro-fevereiro, pp. 38-75, 1992.

[Geoffr92b] Geoffrion, A.M."The SML Language for Structured Modeling: Levels 3 and 4", *Operations Research*, 40:1, janeiro-fevereiro, pp. 38-75, 1992.

[Geoffr93] Geoffrion, A.M."An Reusing Structured Models via Model Integration", *Current research in decision support* techology, IEEE Computer Society Press, p25-55, 1993.

[Geoffr94] Geoffrion, A.M."Structured modeling: survey and future research direction", *ORSA CSTS Newsletter*, 15(1) 10-19, 1994.

[Gorry71] Gorry, G.A., e Morton, M.S.S. "A Framework for Management Information Systems"*Sloan Management Review*, Vol.13, No.1, pp55-70, 1971.

[Hamach93] Hamacher, S., Dejax, P., Lustosa, L., e Hamacher, P, "A Diagram Representation for conceptual Models of Operations Research Problems", *Cahiers d'Etudes et de Recherche No. 93-04A*, Laboratoire Productique Logistique, Ecole Centrale Paris, France, 1993.

[Huber84] Huber, G.P. "Issues in the Design of Group Decision Support Systems."*MIS Quarterly*, 1984.

[Hürlim88] Hürlimann T, Kohlas J."LPL: A Structured Language for Linear Programming Modeling". *OR Spektrum*;10:55-63, 1988.

[Jones92] Jones, C. V., "Attributed Graphs, Graph-Grammars, and Structured Modeling", *Annals of Operations Research*, n.38, p281-324

[Keen78] Keen, P.G.W., Morton, M.S.S. "Decision Support Systems : An Organizational Perspective", *Addison-Wesley Publishers*, Reading Mass., 1978.

[Krishn91] Krishnan, R., "PDM: A Knowlegde-Based Tool for Model Construction", *Decision Support Systems*, Vol.7, No.4, novembro, p301-314, 1991.

[Krishn93] Krishnan, R., "Model management: Survey, future research directions and a bibliography". *ORSA CSTS Newsletter*, 14(1), 1993.

[Ma89] Ma, P.-C., Murphy, F.J., Stohr, E.A, "A Graphics Interface for Linear Programming", *Communications of ACM*, Vol.32, No.8, agosto, p996-1012, 1989.

[Maxima93] MAXIMAL SYSTEMS. MPL user's manual. Arlington, VA,1993.

[Morton71] Morton, M.S.S. "Management Decision Systems: Computer Based Support for Decision Making", Division of Research, Harvard University, Cambridge, 1971.

[Muhann94] Muhanna, W. A. and R. A. Pick, "Meta-Modeling concepts and Tools for Model Management: A System Approach", *Proceeedings of the 23rd Annual Hawaii International Conference on System Sciences*, January, 1990, 328-337.

[Neusta90] Neustadner, L., Geoffrion, A.M., Maturana, S., Tsai, Y., Vicuña, F., "The Design and Implementation of a Prototype Structured Modeling Environment", *Annals of Operations Research.*, Model Management, B. Shetty (Ed.), 1992

[Pereir97] Pereira, D. A. "Extensões à Linguagem SQL para Gerenciamento de Modelos", *Dissertação de Mestrado*, UFMG, Belo Horizonte, Março, 1997.

[Schrag99] Schrage, L. "Optimization Modeling with LINDO 6.1", 5th ed.Chicago: LINDO Systems, 1999.

[Schrag99b] Schrage L. "Optimization Modeling with LINGO 6.0". Chicago: LINDO Systems, 1999.

[Spragu75] Sprague, R.H., Watson, H.J. "Model Management in MIS", *Proceedings Seventh National AIDS*, Cincinatti, p 213-5, 1975.

[Spragu82] Sprague, R.H., Carlson, E.D. "Building Effective Decision Support Systems", *Prentice-Hall*, Englewood Cliffs, New Jersey, 1982.

[Suh95] Suh, C-K., Suh, E-H., Lee, D-M., "Artificial Intelligence Approaches in Model Management Systems: A Survey", *Computers & Industrial Engineering*, Volume 28, Issue 2, Apr 1995, p291-299

[Tsai01] Tsai, Y. "Comparative Analysis of Model Management and Relational Database Management", *Omega*, No 29, pp157-170, 2001.

[Turban98] Turban, E. e Aronson, J. E. "Decision Support Systems and Intelligent Systems". *Prentice Hall*, Englewood Cliffs, New Jersey, 1998.

[Yeo96] Yeo, G. K., "Model Management for Decision Support". *INTERNET*. jul. 1996 (http://www.comp.nus.edu.sg/~yeogk/MM/overview/)