UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# Uma Interface Visual para Modelos de Bancos de Dados Orientados a Objetos com Suporte para Versões

por

JULIANO TONEZER DA SILVA

Dissertação submetida à avaliação como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Clesio Saraiva dos Santos
Orientador

Porto Alegre, outubro de 1998.

**CIP - CATALOGAÇÃO NA PUBLICAÇÃO**

Silva, Juliano Tonezer da

Uma Interface Visual para Modelos de Bancods de Dados Orientados a Objetos com Suporte para Versões / por Juliano Tonezer da Silva. – Porto Alegre: CPGCC da UFRGS, 1998.

91 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 1998. Orientador: Santos, Clesio Saraiva dos.

1. Interfaces visuais. 2. Modelo de dados orientados a objetos. 3. Modelo de versões. 4. Interação homem-computador. 5. Interfaces visuais para modelos de dados orientados a objetos. I. Santos, Clesio Saraiva dos. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Pós-Graduação: Prof. José Carlos Hennemann Ferraz
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenadora do CPGCC: Profa. Carla Maria Dal Sasso Freitas
Bibliotecária – Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

# Agradecimentos

Agradeço a DEUS, por estar sempre conosco, nos guiando e nos dando forças para não desistirmos de nossos objetivos. Obrigado, SENHOR, por ter permitido que eu alcançasse mais esta etapa de minha vida.

Agradeço ao Prof. Clesio Saraiva do Santos, pela sua amizade, compreensão, paciência e disponibilidade constante na orientação deste trabalho. Pelas suas sugestões, críticas construtivas e esclarecimento de dúvidas, que muito contribuíram para a realização deste trabalho. Também gostaria de externar, ao meu Orientador, os meus sinceros agradecimentos pela oportunidade de realizar o mestrado sob sua orientação: Prof. Clesio, Obrigado!

Agradeço ao CPGCC, Curso de Pós-Graduação em Ciência da Computação, da UFRGS, pelos recursos materiais e recursos humanos disponíveis para a realização deste trabalho e para o meu aperfeiçoamento acadêmico e humano. Gostaria de destacar o excelente trabalho que este curso de Pós-Graduação vem realizando junto a esta comunidade científica.

Ao CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico, pelo sustento econômico nestes dois anos de bolsa.

Aos meus queridos Pais: Anselmo e Ana, agradeço pela minha vida e pelos ensinamentos de fé, amor e bondade. Aos meus queridos irmãos: José, Lorita, Cecília, Liliane, Paulo, João e Juliana, agradeço pelo carinho e confiança.

A minha querida namorada, companheira e amiga Carla, os meus sinceros agradecimentos por me suportar nesta reta final da dissertação. *Fofi*, Muito Obrigado!!!

Aos meus colegas de mestrado, que iniciaram comigo em Março de 1996, agradeço pelo companheirismo e amizade. Pelo incentivo nos momentos difíceis e pela partilha dos bons momentos: Galera, valeu, Obrigado.

Ao casal de amigos: Mirto e Glória, agradeço pelo apoio e acolhida neste período em sua residência. Mirto e Glória, Obrigado!

Aos meus parentes e amigos, que de uma forma ou de outra contribuíram para a realização deste trabalho, o meu muito obrigado.

# Sumário

# Lista de Abreviaturas

| | |
|---|---|
| BDO | *Banco de Dados de Objetos* |
| BDOO | *Banco de Dados Orientados a Objetos* |
| CASE | *Computer Aided Software Engineering* |
| CAD | *Computer Aided Design* |
| CPGCC | *Curso de Pós-Graduação em Ciência da Computação da UFRGS* |
| CNPq | *Conselho Nacional de Pesquisa e Desenvolvimento Científico e Tecnológico* |
| DML | *Data Manipulation Language (linguagem de manipulação de dados)* |
| MDOO | *Modelo de Dados Orientado a Objetos* |
| SGBD | *Sistemas de Gerência de Banco de Dados* |
| SQL | *Structured Query Language* |
| UFRGS | *Universidade Federal do Rio Grande do Sul* |
| UML | *Unified Modeling Language* |

# Lista de Figuras

# Lista de Tabelas

# Resumo

Este trabalho apresenta o projeto de uma interface visual para modelos de bancos de dados orientados a objetos, com suporte para versões.

Um requisito importante, não atendido pelas interfaces visuais específicas e genéricas para sistemas orientados a objetos, é a capacidade de definir e manipular versões de um objeto *nos vários níveis da hierarquia de classes* (herança por extensão, adotada pelo modelo de versões [GOL 95]). As interfaces, que manipulam versões, suportam essa característica no nível mais especializado da hierarquia (herança por refinamento, adotada pelos principais SGBDOOs).

Procurando prover a possibilidade do versionamento de objetos *nos vários níveis da hierarquia de classes*, surgiu a motivação para projetar e desenvolver uma interface visual com funcionalidades de interfaces existentes (específicas e genéricas) e que obedeça às características principais dos Modelos de Dados Orientados a Objetos e do Modelo de Versões [GOL 95], seguindo as características recomendadas para interfaces visuais para MDOOs, propostas em [SIL 96].

Foi implementado um protótipo com algumas das características projetadas para o browser de objeto e seu suporte para versões.

**PALAVRAS-CHAVE:** interfaces visuais, modelo de dados orientados a objetos, modelo de versões, interação homem-computador, interfaces visuais para modelos de dados orientados a objetos.

**TITLE:** ‘Visual interface for object-oriented databases with support to the version concept’

# Abstract

This work presents the project of a visual interface for object-oriented databases with support to the version concept.

A fundamental requirement that is not supported by generic and specific visual interfaces, is the capability to define and manipulate object versions in the various levels of the class hierarchy (extention inherience, used by the version model [GOL 95]). The interfaces that manipulate versions, support this feature only in the at specialized level of the class hierarchy (tunning inherience, used by many SGBDOOs).

Therefore, in order to increase the range of visual interfaces for MDOOs and solve the absence of object versioning in the various levels of the inherience hierarchy, it has arised the motivation and the need to design and develop a visual interface wich provides the functionalities of existing interfaces (specific or generic) and that follows the main features of object-oriented data models and the version model [GOL 95] obesrving, the recomended features for MDOOs visual interfaces, proposed in [SIL 96].

A software prototype was developed which performs some of the designed features for the object browser and ics support for versions.

**KEYWORDS:** visual interfaces, object-oriented data model, version model, human-computer interaction, visual interfaces for object-oriented data model.

# 1 Introdução

As interfaces de usuário, em especial as interfaces visuais, têm a preocupação de aprimorar a comunicação e a interação dos usuários com o computador, constituindo-se em elemento essencial e indispensável nos diversos tipos de sistemas computacionais e nas mais diversas áreas: banco de dados (modelos de dados orientados a objetos), computação gráfica, inteligência artificial, simulação, e outras mais.

As interfaces visuais estão, aos poucos, sendo utilizadas em substituição às linguagens de programação tradicionais, devido à constante popularização dos recursos gráficos. As interfaces visuais também proporcionam uma melhor interação do usuário com as informações.

A comunicação (interação) do usuário com as informações pode ser estabelecida através de estilos de interação (ou tipos de diálogo), que podem ser reunidos em grupos, segundo alguns autores [LUC 97b], [OLI 93a], [NIE 93], [GRU 93]: sistemas *batch*; interfaces orientadas à linha; linguagens: linguagem de comando, linguagem de consulta e linguagem natural; sistemas de menus, telas formatadas; formulários e mecanismos de manipulação direta, além das interfaces ditas da próxima geração: mecanismos de navegação em objetos (orientação a objetos), realidade virtual, mecanismos de reconhecimento de voz, entre outras a surgir.

A tarefa de proporcionar a interação do usuário com as informações, através das interfaces, não é um processo simples, pois muitos aspectos contribuem para aumentar essa dificuldade. A interface deve satisfazer os diferentes tipos de usuários: programadores de aplicações, usuários sofisticados, usuários especializados e usuários ingênuos; e esses com níveis de conhecimento diferenciados do sistema: usuários novos, infreqüentes e experientes [OLI 93a], [KOR 93]. Outro aspecto é que o projeto de uma interface, idealmente, envolve várias áreas, que contribuem mais ou menos que outras, dependendo da natureza da interface [DOW 91]. Podemos citar algumas: Ciência da Computação, Engenharia, Ergonomia, Psicologia, Sociologia, Lingüística, Desenho Gráfico e Tipografia, e Inteligência Artificial [SJO 96], [DOW 91].

As interfaces visuais na área de sistemas de banco de dados estão possibilitando aos seus usuários o acesso às informações (objetos), de um modo cada vez mais direto e intuitivo, tornando esse processo de comunicação usuário - banco de dados, mais agradável, simples e eficaz.

Poucas das interfaces específicas e genéricas, comerciais ou experimentais, disponíveis para Modelos de Dados Orientados a Objetos, permitem definir e manipular versões de objetos [VIV 96], [OLI 93a, OLI 94], [CAR 96, CAR 95], [BUT 91]. E as interfaces que suportam versões, via de regra, permitem somente o versionamento nos níveis mais especializados das hierarquias de classes [$O_2$ 95], ODE [AGR 90].

Não existe um MDOO padrão, mas há um certo consenso em relação às principais características e aspectos que qualificam um MDOO. Dois documentos, entre outros,

que descrevem esses aspectos podem ser encontrados em [BER 93] e no manifesto [ATK 89]. Esses documentos são citados em várias referências.

Um aspecto dos MDOOs, que é importante ser destacado, é o tipo de herança adotada pelo modelo de versões, pois isto influenciou diretamente o projeto da interface. Em [BIL 90] são descritos dois tipos de herança: por refinamento e por extensão. O modelo de versões, adotado neste trabalho, [GOL 95] baseia-se na herança por extensão, que está relacionada com a idéia de protótipos e aparece em modelos de dados como o Modelo E [SAN 86, SAN 89].

Na herança por extensão, cada objeto em uma *subclasse* (p.e. *O2*), possui um objeto associado em sua *superclasse* (p.e. *O1*). No modelo de versões, *O1* é denominado ascendente e *O2* é chamado descendente de *O1*. Em [BIL 90], *O1* é denominado protótipo e *O2* (descendente) é uma extensão do protótipo.

Na herança por refinamento, uma entidade do mundo real é modelada como um único objeto, que é instância da classe mais especializada na hierarquia de classes a qual pertence. Todos os atributos aparecem na *subclasse* mais especializada. Os principais SGBDOOs existentes adotam a herança por refinamento.

O modelo de versões é uma extensão para bancos de dados orientados a objetos e está descrito em [GOL 95]. Uma versão é a descrição de um objeto em um determinado momento de tempo, ou sob um determinado ponto de vista, cujo registro é importante para a aplicação. No modelo de versões, cada versão representa um estado identificável de um objeto, considerado pelo usuário como semanticamente significativo, devendo ser tratada como um objeto no modelo de dados.

O objetivo deste trabalho de dissertação é projetar uma interface visual para modelos de bancos de dados orientados a objetos e que, principalmente, suporte diferentes versões de um objeto.

Com a herança por extensão, surge a presença de versões nos diferentes níveis da hierarquia de classes, que pode resultar em inúmeras combinações de versões, tornando difícil para o usuário tratar diretamente cada uma das possíveis combinações. É importante, pois, que a interface apresente mecanismos adequados para o tratamento de versões nos vários níveis da hierarquia de classes.

Neste contexto, ficam estabelecidas correspondências (mapeamentos) entre versões de um objeto em uma classe e versões de seu objeto ascendente na superclasse. A correspondência também estabelece uma restrição de integridade, que é especificada quando da definição do esquema (relacionamento de herança entre uma classe e sua superclasse) e deve ser mantida pelo sistema.

Além das correspondências, também existem as referências a componentes que podem ser estáticas ou dinâmicas. Com a referência dinâmica, surge o conceito de configuração. Considerando um objeto composto, em que os componentes podem ser objetos versionados, uma configuração associa exatamente uma versão para cada um desses componentes [GOL 95].

O layout de janelas da interface foi projetado em conformidade com o padrão de interface da Microsoft®, descrito no guia de recomendação [MS 97]. A interface é basicamente composta por uma janela principal e janelas (formulários), denominados de *browsers*, onde podem ser visualizadas as informações apresentadas pela interface.

As janelas de informações foram agrupadas nos seguintes *browsers*: (1) *browser de banco de dados:* apresenta, através de ícones, os bancos de dados que, selecionados, inicia-se uma sessão de trabalho; (2) *browser de esquema:* apresenta o esquema do banco de dados ativo, mostrando a hierarquia de classes; (3) *browser de classes:* apresenta as informações de uma das classes do esquema por vez, mostrando os atributos e métodos desta classe, textualmente; (4) *browser de objeto:* (suporte para versões de objetos) apresenta os objetos da classe e as versões dos objetos versionados; e (5) *browser de consulta:* permite a consulta aos objetos da classe e as versões dos objetos.

A janela do browser de objeto foi dividida em múltiplas regiões (viewports): (1) *viewport* Contexto; (2) *viewport* Superclasses; (3) *viewport S*ubclasses; (4) *viewport* Instância; e (5) *viewport* Grafo. As três primeiras *viewports* apresentam informações do esquema, com âncoras para as classes relacionadas. Na *viewport* Instância são visualizados os atributos do objeto corrente ou da versão corrente; e na *viewport* Grafo é mostrado o grafo do objeto versionado corrente, com as suas versões e as correspondências entre as versões.

A presente dissertação é composta por seis capítulos, organizados da seguinte forma: O capítulo dois apresenta um estudo de interfaces de usuário, destacando a interação Homem-Computador e os seus estilos de interação. Também são descritos brevemente o projeto e a avaliação de interfaces de usuário. No capítulo três são descritas algumas interfaces visuais para modelos de dados orientados a objeto, apresentando suas principais características conforme a afinidade dessas com os *browsers*, sugeridos em [SIL 96]. No capítulo quatro é descrito o projeto da Interface, apresentando as suas características. No capítulo cinco é apresentada a implementação de um protótipo com algumas das características descritas no projeto da interface. A funcionalidade explorada no protótipo foi o *Browser de Objeto* e o seu suporte para versões. Nesse capítulo também está descrito um estudo de caso realizado com o protótipo. No capítulo seis são apresentadas conclusões a respeito do projeto da interface, do protótipo e de interfaces de usuário.

# 2 Interfaces de Usuário

Para Levy [LEV 94], *a interface é um elemento imprescindível para a aceitação de um sistema interativo por parte do usuário*, pois, muitas vezes, o usuário prefere uma interface mais "amigável" a um sistema com mais desempenho e recursos.

## 2.1 Interação Homem-Computador[1]

Inserida no contexto de Interfaces de usuário, a Interação Homem-Computador desperta o interesse de muitos pesquisadores em estudar esse assunto, de extrema importância nos sistemas de computação, e de rápida evolução. Como exemplo disso, podemos citar a realização de conferências internacionais, específicas ou pertinentes, que discutem esse tema: APCHI (*Asia-Pacific Conference on Human Computer Interaction*), CHI (*Conference on Human Factors in Computing Systems*), *International Conference on Human-Computer Interaction*, *British Computer Society HCI: Human-Computer Interaction*, entre outras. Também há periódicos que apresentam trabalhos nessa área: *ACM SIGCHI Bulletin*, *ACM TOCHI*, *ACM Interactions*, *Human-Computer Interactions* (jornal), *International Journal of Human-Computer Interaction*, entre outros. Uma biblioteca eletrônica, a *HCI Bibliografy Project*, mantém uma base de dados com inúmeras referências sobre o tema. No HCIL, *Human-Computer Interaction Laboratory*[2] da Universidade de *Maryland*, encontra-se uma grande quantidade de artigos e relatórios técnicos relacionados com o assunto. Um documento que traz uma boa orientação inicial para uma pesquisa bibliográfica relacionada à interação homem-computador é encontrado em [LUC 97a]. Também existe um cadastro eletrônico dos pesquisadores de HCI brasileiros, com o nome de sua instituição e a área de estudo em HCI[3].

A interação Homem-Computador pode ser definida como: "*Uma área voltada para o desenvolvimento de sistemas computacionais, que dão suporte a pessoas, para que possam realizar suas atividades de forma produtiva e segura.*" [PRE 94]

Faz-se necessário entender a diferença entre interface e interação (comunicação) Homem-Computador. Interação se refere a tudo o que acontece no processo de comunicação do usuário com o computador. A Interface é o mecanismo (hardware + software) que proporciona essa comunicação [LUC 97b].

Nielson [NIE 92] relaciona a Interação Homem-Computador, nos sistemas computacionais, com esses cinco atributos, associados à usabilidade (facilidade de uso e aprendizado): (1) a*prendizagem:* o sistema deve ser fácil de aprender; (2) *eficiência: o* sistema deve ser eficiente para usar. Após o usuário ter aprendido o sistema, um alto nível de produtividade é esperado; (3) *memorabilidade:* as funcionalidades do sistema devem ser fáceis de lembrar. Usuários casuais devem ser capazes de retornar ao sistema depois de algum período sem usá-lo, sem precisar reestudar novamente sobre esse; (4) *erros:* o sistema deve ter um baixo índice de erros e esses erros devem ser

[1] Human-Computer Interaction (HCI).
[2] http://www.cs.umd.edu/project/hcil/ research/.
[3] http://www.inf.puc-rio.br/~raquel/hci_por.html.

facilmente recuperados. Erros catastróficos não podem ocorrer; (5) *satisfação:* o sistema deve ser agradável ao usuário. Essa satisfação é subjetiva, relacionada com o uso do sistema pelo usuário.

Lucena [LUC 97b] cita algumas características que normalmente aparecem em interfaces amigáveis (*user-friendly*): (1) Facilidade de usar e aprender*; (2)* Taxa de erro mínima; (3) Recordação rápida; (4) Atrativo. A palavra usabilidade também é utilizada em alguns contextos semelhantes, sendo que não há uma definição precisa para esses termos.

## 2.2 Estilos de interação homem-computador

Um estilo de interação, ou tipo de diálogo, refere-se à forma através da qual ocorre uma interação (comunicação, diálogo) entre o usuário e o computador. Em geral, vários estilos de interação são utilizados em uma mesma interface [LUC 97b]

Nielson [NIE 93] reúne os estilos de interação em cinco categorias: (1) *Sistemas Batch*; (2) Interfaces *line-oriented*: diálogos pergunta-respostas e diálogos de linha de comando; (3) Interfaces *full-screens*: preenchimento de formulários e sistemas de menus; (4) Manipulação Direta; (5) Próxima Geração de Interfaces.

Em Oliveira [OLI 93a], os estilos de interação são reunidos, de um modo geral, em quatro grupos: (1) linguagens: linguagem de comando, linguagem de consulta, linguagem natural; (2) telas formatadas; (3) menus; (4) mecanismos de manipulação direta.

Lucena [LUC 97b] descreve vários estilos de interação, não seguindo nenhum agrupamento: WYSIWYG; Linguagem de comandos; Linguagem natural; Manipulação direta; Reação por inferência; Icônico; Menus; Preenchimento de formulários; e Caixas.

Em Grundin [GRU 93], tem-se uma noção de como as interfaces evoluíram desde as primeiras máquinas calculadoras: (1) *década de 40:* a interação era física (conectores, fios, chaves e outras peças do próprio hardware); (2) *década de 50:* interfaces eletromecânicas: teletipos, leitoras e perfuradoras de cartão, painéis luminosos e mostradores alfanuméricos; (3) *década de 60:* tubos de raios catódicos (CRTs); (4) *década de 70:* terminais gráficos (*bitmapped*), primeiras experiências com dispositivos apontadores e ícones; (5) *década de 80:* interfaces gráficas: interfaces WIMP[1] e a manipulação direta;

*Atualmente, na década de 1990, presencia-se o resultado da explosão das interfaces gráficas, que agora exploram caminhos que há décadas atrás seriam insólitos: a multimídia e a realidade virtual.* [CHA 96]

Myers [MYE 96] descreve a história da tecnologia da interação Homem-Computador em quatros grupos:

[1] WIMP: é o acrônimo de *Window*, *Icon*, *Menu* and *Pointing device*

• *Estilos de interação*: manipulação direta de objetos gráficos, o mouse (dispositivos de apontadores) e *windows* (janelas);

• *Tipos de Aplicações:* programas de desenho (*drawing programs*), editores de texto (*text editing*), planilhas (*spreadsheets*), hipertexto (*hypertext*), aplicativos CAD (*Computer Aided Design*) e jogos (vídeo games);

• *Áreas ativas:* Reconhecimento de gestos (*gesture recognition*), multimídia (*multi-media*), 3-D, realidade virtual (*virtual reality*), trabalho em grupo (*computer supported cooperative work*), linguagem natural (*natural language*);

• *Software e arquitetura:* UIMSs[1] e Toolkits, construtores de interfaces e arquitetura de componentes (OLE, Aplet's, ...). A figura 2.1 apresenta uma linha de tempo com o início aproximado do aparecimento das tecnologias dentro das: Universidades, Corporações e como produtos comerciais.

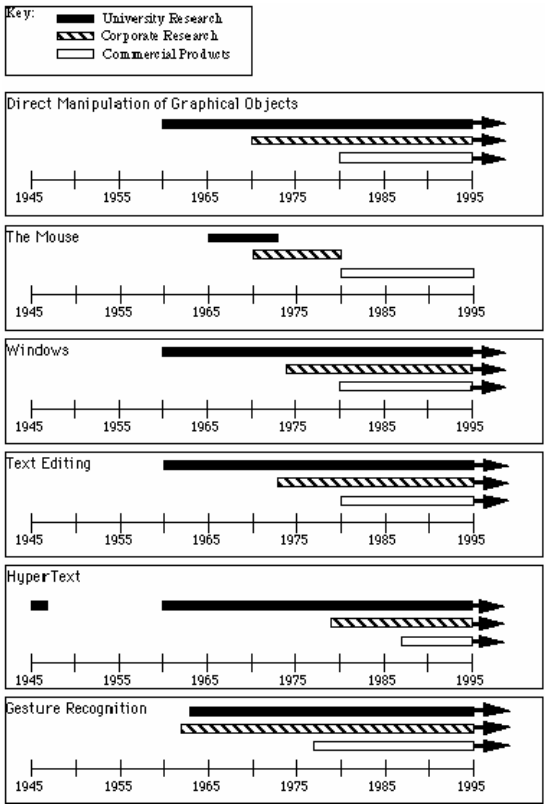



FIGURA 2.1- Linha do tempo aproximada [MEY 96]

Uma forma interessante para descrever os estilos de interação é apresentá-los em sua ordem cronológica. Nielson [NIE 93] divide a *história da interação* em cinco níveis[2]. Utilizando-se dessa divisão citada por Nielson, serão descritos os demais estilos de interação Homem-Computador: (1) *nível zero:* sistemas batch; (2) *nível um:*

[1] User Interface Management System

[2] o autor utiliza o termo *dimensional* (*zero-dimensional*, *one-dimensional*, ...), que significa mensurável

interfaces *line-oriented*, linguagem de comando; (3) *nível dois:* linguagem de consulta e linguagem natural; menus; telas formatadas e preenchimento de formulários; (4) *nível três:* manipulação direta, icônico e reação por inferência; (5) *próxima geração de interfaces:* técnicas de realidade virtual; mecanismos de reconhecimento de voz; mecanismos de navegação em objetos de banco de dados.

### 2.2.1 Sistemas Batch

Essa primeira geração de interfaces não foi interativa. O usuário especificava todos os comandos previamente em um arquivo batch (em lote). A interação entre os usuários e o sistema se restringia no momento em que o arquivo batch era submetido ao sistema.

Nielson cita que a grande vantagem dos sistemas batch foi, e ainda é, que os processos podem ficar executando as instruções sem a participação do usuário.

Recentemente, os sistemas batch foram redescobertos na forma de sistemas de correio eletrônico, ftps (file transfer protocolo) que é o protocolo utilizado para transferência de arquivos entre máquinas na Internet.

### 2.2.2 Interfaces através de comandos

O usuário podia interagir com o computador através da única linha de ação, a linha de comando. Depois que o usuário digitava a instrução de entrada e pressionava a tecla ENTER, sua entrada não podia mais ser modificada. Essas interfaces não permitiam ao usuário mover-se na tela [NIE 93].

Dentro da categoria de interfaces, através de comandos, estão diálogos pergunta-resposta e diálogos de linha de comando.

### 2.2.3 Linguagens

As linguagens formam o mais utilizado estilo de interação Homem-Computador. Ele divide-as em três grupos distintos: linguagem de comando, linguagem de consulta e linguagem natural [OLI 93a].

#### 2.2.3.1 Linguagem de Comando

As estratégias desenvolvidas pelos estudiosos de linguagens de comandos para dar nomes adequados aos comandos influenciaram fortemente outros estilos de interação, como os sistemas de menus. Podendo-se estender e utilizar os princípios de projeto de uma linguagem de comando em qualquer estilo de interação onde nomes sejam usados para representar opções ou ações.

As linguagens de comando são mais indicadas para usuários que possuem um bom conhecimento do sistema em que eles atuam. Nesse caso, as linguagens de comando oferecem um modo rápido e eficiente de acesso à funcionalidade do sistema. É

provável que essas permaneçam como o estilo de diálogo preferido de um grande número de usuários experientes.

As linguagens de comando, para usuários novatos ou casuais, são difíceis de aprender, requerem um esforço razoável de prática e memorização e levam a altas taxas de erros. As linguagens de comando, geralmente, são empregadas por usuários experientes em uma aplicação. Elas fornecem um acesso rápido aos recursos fornecidos pelo sistema, mas exigem uma boa memorização [LUC 97b].

#### 2.2.3.2 Linguagem de Consulta

São linguagens de comando que têm o propósito especial de recuperar informações em Sistemas de Banco de Dados. Como o próprio nome diz, são linguagens que executam *consulta* aos bancos de dados.

As linguagens de consulta diferem em três aspectos: paradigma de programação, modelo de dados e forma sintática. O aspecto *paradigma de programação* identifica se a consulta é procedural ou não [OLI 93]. Em uma linguagem procedural, o usuário deve especificar o procedimento de busca, enquanto que na linguagem de consulta não-procedural precisa apenas informar as características da consulta, deixando que o sistema encontre um algoritmo eficiente para buscar os dados armazenados. No aspecto "*modelos de dados*", a linguagem de consulta deve refletir as estruturas usadas no modelo de dados empregado pelo SGBD. A *forma sintática* pode ser *linear* ou *tabular*. Duas linguagens de consultas bem populares que são exemplos dessas sintaxes são SQL e QBE, respectivamente.

#### 2.2.3.3 Linguagem Natural

A linguagem natural busca estabelecer um estilo de interação que se assemelhe à linguagem utilizada pelos homens para se comunicarem.

Uma interface que implementasse, de maneira completa, uma linguagem natural possibilitaria que pessoas leigas pudessem utilizar os serviços de um sistema de computador, sem a necessidade de qualquer tipo de treinamento prévio [OLI 93].

A linguagem natural é uma extensão da linguagem de comando, onde o usuário não fica limitado a usar um vocabulário exíguo (escasso) de palavras e uma sintaxe rígida para expressar os comandos [LUC 97b].

A necessidade constante da recuperação automática de textos (TR), também conhecida como recuperação de documentos (DR), tem despertado a atenção de pesquisadores no processamento de linguagem natural [LEW 96].

Projetos desenvolvidos em vários sistemas de processamento de linguagem natural (NLP) são caros e complexos, mas o resultado dos dados é valioso (inestimável) [KIN 96].

### 2.2.4 Sistemas de Menus

Um menu pode ser definido como um conjunto de opções apresentadas para o usuário, onde a seleção de uma opção resulta na modificação do estado da interface. As opções de um menu podem ser representadas por números, palavras, ícones, entre outras formas [OLI 93a].

Para Baecker e Buxton, um menu é uma exibição de alternativas, em que uma opção ou valor é discriminado ou selecionado em cada ciclo. Os sistemas de menus reduzem a necessidade de memorização e limitam o conjunto de opções [LUC 97b].

### 2.2.5 Telas Formatadas e Preenchimento de Formulários

Esses tipos de diálogos referem-se ao tipo de interação onde o usuário se restringe a completar os campos livres de telas pré-formatadas, num processo análogo ao de preenchimento de formulários de papel [OLI 93].

Esses estilos de interação estão sendo substituídos em muitos sistemas computacionais por interfaces mais eficientes como, por exemplo, leitoras de código de barra e Assistentes Digitais Pessoais (PDAs - Personal Digital Assistants).

### 2.2.6 Manipulação Direta

O principal estilo de interação homem-computador é a manipulação direta em que, através de ações físicas (como selecionar, arrastar e soltar), o usuário manipula graficamente os objetos na tela. O paradigma de manipulação direta de objetos foi introduzido por Ben Shneiderman [SHN 83].

Segundo Shneiderman, as principais características das interfaces de manipulação direta são:

- a representação contínua de objetos de interesse;
- ações físicas ou rótulos de botões em vez de sintaxes complexas;
- a utilização de operações rápidas, incrementais e reversíveis, cujo efeito nos objetos de interesse é imediatamente visível pelo usuário;
- a abordagem gradual para aprendizagem, permitindo o uso de sistema com um mínimo de conhecimentos, e a expansão da capacitação do usuário através do próprio uso do sistema.

Shneiderman associa a manipulação direta com as seguintes idéias centrais: (1) visibilidade do objeto de interesse; (2) ações rápidas e reversíveis; (3) visualização imediata do resultado; (4) substituição dos estilos de interação, como os de linguagem de comandos e menus pela manipulação direta da representação do objeto de interesse.

Sjoerd [SJO 96] destaca alguns benefícios e problemas decorrentes da manipulação direta:

- *Benefícios da manipulação direta:*

- novos usuários podem aprender funcionalidades básicas rapidamente, através da demonstração por um usuário mais experiente;
- usuários experientes podem trabalhar rapidamente, obtendo resultados em uma grande faixa de tarefas;
- usuários ocasionais podem reter os conceitos operacionais devido à não complexidade da sintaxe e podem lembrá-los sempre;
- resulta numa sintaxe menor e reduz o índice de erros. Mensagens de erros são raramente necessárias;
- usuários podem, imediatamente, ver se suas ações estão mais longe de seus objetivos e se as ações estão sendo produtivas. Eles podem simplesmente mudar a direção de sua atividade;
- usuários experientes reduzem sua ansiedade, porque o sistema fica mais compreensível e porque as ações podem ser revertidas facilmente.
- usuários ganham confiança e domínio, porque eles são os iniciadores da ação: eles sentem o controle e a resposta do sistema.

- *Problemas com a manipulação direta:*

  - em geral, interfaces de manipulação direta requerem mais recursos do sistema;
  - uma operação repetitiva é, provavelmente, melhor, se feita via um script ou macro;
  - interfaces de manipulação direta têm problemas de exatidão na representação de ações físicas por objetos gráficos;
  - um problema mais fundamental surge a partir do fato de que a manipulação direta requer um substancial conhecimento da palavra.

#### 2.2.6.1 Metáforas

Carroll et. al., cita três estágios no raciocínio de metáforas, cada um associado com diferentes tipos de atividades cognitivas: (1) instanciação; (2) elaboração; e (3) consolidação.

Carroll também destaca que há quatro passos para o projeto de interfaces, usando metáforas: (1) identificar as metáforas candidatas; (2) detalhar o software/metáfora, comparando com o respectivo cenário do usuário; (3) identificar os inevitáveis “mismatches” e suas implicações; e (4) a identificação das estratégias de projeto para auxiliar os usuários a gerenciar os “mismatches”.

### 2.2.7 Próxima Geração de Interfaces

#### 2.2.7.1 Reconhecimento de Voz

Uma interface de reconhecimento de voz deve tratar de seis propriedades básicas, que são: (1) Fala dependente X Fala independente; (2) Palavras discretas X Palavras conectadas X Fala contínua; (3) Tempo real X Processamento offline; (4)

Vocabulário grande X Vocabulário pequeno; (5) Gramática ampla X Gramática restrita; (6) Reconhecimento de voz X Reconhecimento de Locutor X Entendimento de Voz.

Um *site* na Internet que apresenta muitas publicações na área de Linguagem Humana (reconhecimento de voz), incluindo: livros, artigos, relatórios técnicos e teses é o do *Center for Spoken Language Understanding* de *Oregon Graduate Institute of Science and Technology*, disponível em **http://www.cse.ogi.edu/CSLU/publications/**.

#### 2.2.7.2 Mecanismos de Navegação

Oliveira descreve o termo navegação como "*o processo de visualização seqüencial de informações de um determinado tipo*".

Várias Interfaces incluem essa funcionalidade em suas características. Entre essas podemos citar: (1) a interface do sistema GOODIES[1] [OLI 93a, OLI 93b, OLI 93c, OLI 93d, OLI 94]; (2) a interface de PESTO[2] [CAR 96, CAR 95]; (3) ObjectStore Inspector, um browser visual para o SGBDOO ObjectStore [VIV 96]; (4) OdeView, a interface visual do SGBDOO ODE [AGR 90], entre outras. Essas interfaces para modelo de dados orientados a objeto estão descritas no capítulo 3.

#### 2.2.7.3 Realidade Virtual

Sclovsky [SCL 94] descreve a realidade virtual como: "*um novo paradigma de interface homem-máquina, que constitui um salto qualitativo em relação a uma interface gráfica*".

## 2.3 Projeto de Interfaces

*Corrigir modelos e especificações de projeto é uma atividade bem menos onerosa do que os seus efeitos correspondentes sobre objetos reais* [LUC 97b]. Lucena descreve o projeto de interação como uma atividade que define o *look and feel*[3] de uma interface, ou seja, o comportamento e a apresentação de uma interface.

O projeto de uma interface envolve o conhecimento de várias áreas, fora do âmbito da ciência da computação. Muitas interfaces são desenvolvidas somente por especialistas em computação, que utilizam apenas seus conhecimentos e experiências empíricas dessa área.

Interfaces projetadas sem os conhecimentos especializados das áreas que devem estar envolvidas nesse projeto acabam forçando o usuário a se adaptar ao sistema para

---

[1]GOODIES é um acrônimo para **G**raphical **O**bject **O**riented **D**atabase **I**nterface with **E**xtended **S**ynchronism (Interface Gráfica para Banco de Dados Orientados a Objeto com Sincronismo Essendido).
[2]PESTO: An Integrated Query/Browser for Object Databases. PESTO é um acrônimo para **P**ortable **E**xplorer of **ST**ructured **O**bjects.
[3] O termo *look and feel* compreende tanto a aparência quanto a dinâmica de interação de uma interface.

compensar as suas deficiências [DOW 91]. Dependendo da natureza da interface, algumas áreas podem contribuir mais que outras [DOW 91]. Downton descreve a contribuição, no desenvolvimento de uma interface, nas seguintes áreas: (1) *Ciência da Computação:* fornece a estrutura tecnológica para facilitar o projeto e implementação da interface; (2) *Psicologia:* preocupa-se com o entendimento do comportamento humano, percepção, cognição e outros; (3) *Ergonomia:* envolve-se com os aspectos físicos da adaptação das máquinas para o uso humano; (4) *Lingüística:* é o estudo de linguagens. Idealmente, mensagens de erros devem ser definidas por especialistas nessa área; (5) *Sociologia:* preocupa-se com o impacto dos sistemas interativos na estrutura da sociedade; (6) *Projeto Visual:* as habilidades estéticas dessas áreas são importantes à medida que tornam ”bonita” e “atraente” a apresentação de uma interface aos olhos do usuário;

A interação Homem-Computador é um campo que envolve várias áreas, considerando-a como uma área multidisciplinar. A figura 2.2 mostra as áreas que contribuem para a interação Homem-Computador [SJO 96].

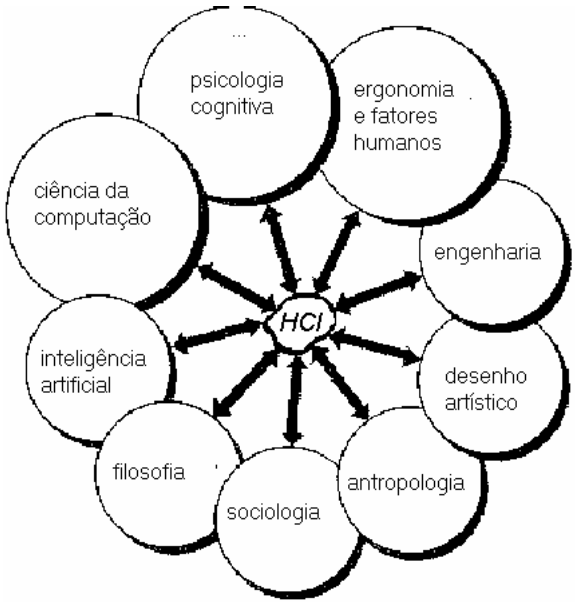



FIGURA 2.2- Áreas que contribuem no projeto de uma interface [SJO 96]

Muitos modelos foram propostos: (1) ciclo de vida cascata; (2) ciclo de vida espiral [BOE 88]; (3) ciclo de vida estrela [PER 89]; buscando aprimorar cada vez mais a qualidade da interação entre os usuários e o computador, através das interfaces. Mas, atualmente, *inexistem regras de ouro que, uma vez seguidas, conduzem à usabilidade esperada para um sistema* [LUC 97b]. Lucena acrescenta que um modelo pode oferecer uma abordagem organizada para o projeto de uma interface, mas não assegura que os resultados desejados serão alcançados.

Nos modelos mais recentes, há um consenso quanto à **iteração**, *que é vista como o único meio de atingir resultados satisfatórios através da contínua avaliação do projeto, especialmente por parte de usuários típicos* [LUC 97b].

## 2.4 Métodos de Avaliação de Interfaces

Não há um consenso de como projetar e avaliar adequadamente as interfaces Homem-Computador [MAT 95]. Mas algumas ferramentas têm sido utilizadas: (1) normas; (2) guias de recomendações; (3) guias de estilos; e (4) técnicas de avaliação; e (5) critérios ergonômicos.

### 2.4.1 Normas

Para Hix, as *"normas são documentos oficiais disponíveis publicamente que fornecem requisitos para o projeto e avaliação de interação"*;

### 2.4.2 Guias de recomendação

Os guias de recomendação *"são publicações que agrupam recomendações derivadas empiricamente ou validadas e que foram compiladas sem a preocupação com a aplicação sistemática das mesmas"* [MAT 95].

### 2.4.3 Guias de estilos

São documentos que contêm orientações de como utilizar os estilos de interação Homem-Computador. Têm o objetivo de documentar o padrão para produtos comerciais de empresas específicas. Podemos citar alguns: (1) Commom User Access - CUA [IBM 92]; (2) Motif, OSF (Open Software Foundation); (3) Microsoft Windows [MS 97]; (4) Sun Microsystems [SUN 90]; (5) Apple Computer [AC 93].

### 2.4.4 Técnicas e avaliação

Jeffries [JEF 91] descreve quatro técnicas para a avaliação de interfaces: (1) avaliação heurística, (2) teste de usabilidade, (3) conformidade com os guias: recomendações e estilos; e (4) exploração cognitiva;

- *avaliação heurística:* *na avaliação heurística especialistas em interfaces estudam em profundidade e observam as propriedades que eles, por experiência, sabem que apresentarão problemas de usabilidade*;

- *teste de usabilidade:* os produtos são testados com a participação de usuários durante o desenvolvimento do projeto. *A facilidade de aprendizagem de software não pode ser julgada por avaliadores que estejam intimamente envolvidos no seu desenvolvimento* [POW 90].

- *conformidade com os guias:* a conformidade com os guias fornecem aos avaliadores recomendações específicas sobre o projeto de uma interface: por exemplo, como as opções deveriam estar arranjadas em um menu, etc.

### 2.4.5 Critérios ergonômicos

Bastien and Scapin [BAS 93] estabeleceram oito critérios ergonômicos que podem ser utilizados para a avaliação de interfaces Homem-Computador. Os critérios propostos são: (1) *Condução:* Presteza, Agrupamento/Distinção de ítens, Feedback Imediato e Legibilidade; (2) *Carga de Trabalho:* Brevidade e Densidade Informacional; (3) *Controle Explícito:* Ações Explícitas do Usuário e Controle do Usuário; (4) *Adaptabilidade:* Flexibilidade e Consideração da Experiência do Usuário; (5) *Gestão de erros:* Proteção contra os Erros, Qualidade das Mensagens de Erro, e Correção dos Erros; (6) *Homogeneidade/ Coerência*; (7) *Significado dos códigos e denominações*; e (8) *Compatibilidade*.

- *Condução: refere-se aos meios disponíveis para aconselhar, orientar, informar e conduzir o usuário na interação com o computador (mensagens, alarmes, rótulos, etc.)*

- *Carga de Trabalho: diz respeito a todos os elementos da interface que têm um papel importante na redução da carga perceptiva e cognitiva do usuário, e no aumento da eficiência do diálogo.*

- *Controle Explícito: diz respeito tanto ao processamento explícito pelo sistema das ações do usuário, quanto do controle que os usuários têm sobre o processamento de suas ações pelo sistema.*

- *Adaptabilidade: refere-se à capacidade do sistema de reagir conforme o contexto e conforme as necessidades e preferências do usuário.*

- *Gestão de erros: diz respeito a todos os mecanismos que permitem evitar ou reduzir a ocorrência de erros e, quando eles ocorrerem, o sistema deve favorecer a sua correção. São considerados erros: entrada de dados incorretos, entradas com formatos inadequados, entradas de comandos com sintaxes incorretas, etc.*

- *Homogeneidade/Coerência: refere-se à forma na qual as escolhas na concepção da interface (códigos, denominações, formatos, procedimentos, etc.) são conservadas idênticas em contextos idênticos, e diferentes em contextos diferentes.*

- *Significado dos códigos e denominações: refere-se à adequação entre o objeto ou a informação apresentada ou solicitada, e a sua referência. Códigos e denominações significativos possuem uma forte relação semântica com seu referente.*

- *<u>Compatibilidade:</u> diz respeito, de uma parte, à relação entre as características do usuário (memória, percepção, hábitos, competências, idade, expectativas, etc.) e as características da tarefa. De outra parte, refere-se à organização das saídas, das entradas e do diálogo de uma dada aplicação. A Compatibilidade também diz respeito ao grau de similaridade entre diferentes ambientes e aplicações.*

Observou-se que o projeto de uma interface para modelos de bancos de dados orientados a objetos e com suporte para versões requer a colaboração de especialistas em interface homem-computador, além da contribuição de profissionais de outras áreas e, principalmente, a participação de futuros usuários da interface. Também percebeu-se a necessidade de submeter o projeto de uma interface a critérios de avaliação para verificar se a interface será "usável" (usabilidade), além da importância da utilização de técnicas que auxiliem na obtenção dos resultados esperados, previamente projetados.

Caso a interface seja projetada para uma plataforma específica, é importante ressaltar que se deve utilizar um guia de estilo para essa plataforma específica. Esses documentos contêm orientações de como utilizar os estilos de interação Homem-Computador, além de documentar o padrão para produtos comerciais da empresa específica.

# 3 Interfaces Visuais para Modelos Bancos de Dados Orientados a Objeto

Este capítulo apresenta algumas interfaces visuais para modelos de dados orientados a objetos: *ObjectStore Inspector*, um browser visual para o SGBDOO ObjectStore Object [VIV 96], OdeView, a interface visual do ODE [AGR 90], a interface gráfica do $O_2$ [$O_2$ 95], GOODIES[1] [OLI 93, OLI 94], PESTO[2] [CAR 96, CAR 95].

## 3.1 A Navegação

Uma funcionalidade favorecida pelo fato da interface ser visual é a navegação. Essa característica mostra-se útil quando há necessidade de manipular as informações do banco de dados, passo a passo. Por exemplo, quando se deseja visualizar as diferentes versões de um objeto versionado, fica mais fácil e intuitivo navegar sobre as versões, percorrendo-as uma a uma.

A interface ObjectStore Inspector, um browser visual para o SGBDOO ObjectStore [VIV 96], permite ao usuário navegar tanto pelo esquema do banco de dados, como pelos objetos (instâncias) e seus relacionamentos. Com um simples *click*, o usuário pode navegar entre as instâncias relacionadas. O caminho de navegação percorrido é armazenado como uma árvore, para posterior análise e compreensão.

Na interface visual do ODE, OdeView [AGR 90], pode-se examinar a definição das classes e navegar pelos objetos. OdeView tem uma janela de apresentação de objetos que é composta de três partes: (1) *o painel de controle:* onde estão os botões para navegação através dos objetos (*reset*, *next*, *previous*); (2) *o painel de objetos:* onde estão os botões para visualização dos objetos; (3) *o painel de informações*: onde são visualizados os objetos. OdeView também permite a navegação sobre objetos complexos e a navegação sincronizada.

A interface GOODIES [OLI 93] apresenta os seguintes mecanismos de navegação: (1) *navegação no esquema:* o usuário pode navegar pelo esquema do banco de dados, selecionando classes (nas listas de superclasses e subclasses), ou métodos (na lista de métodos); (2) *navegação pelos objetos:* inicia-se com a seleção de um objeto na lista de objetos, onde o usuário pode trabalhar com uma ou várias instâncias ao mesmo tempo. A navegação se dá através das operações de seqüenciamento; (3) *facilidade de consulta:* essa é uma capacidade adicional que torna o mecanismo de navegação de GOODIES mais poderoso, podendo ser considerado como um processo simplificado de navegação e consulta aos objetos do banco de dados.

---

[1] GOODIES é um acrônimo para **G**raphical **O**bject-**O**riented **D**atabase **I**nterface with **E**xtended **S**ynchronism (Interface Gráfica para Banco de Dados Orientados a Objeto com sincronismo Essendido).

[2] PESTO: An Integrated Query/Browser for Object Databases. PESTO é um acrônimo para **P**ortable **E**xplorer of **ST**ructured **O**bjects.

A interface PESTO [CAR 96, CAR 95] permite ao usuário navegar pelos objetos e pelos relacionamentos que existem entre os objetos. PESTO também permite a navegação sincronizada e a navegação entre objetos complexos, além dos mecanismos de consulta.

## 3.2 Os browsers

Em [SIL 96], estão descritas algumas características recomendadas para interfaces visuais para modelos de dados orientados a objetos. Essas características foram agrupadas em *browsers*, conforme o nível de abstração da informação a ser visualizada. Os seguintes *browsers* foram sugeridos: (1) *browser de banco de dados*; (2) *browser de esquema*; (3) *browser de classe*; (4) *browser de objeto*; e (5) *browser de consulta*.

As interfaces visuais serão apresentadas no decorrer deste capítulo, conforme as afinidades de suas características com os *browsers* sugeridos. A proposta da interface está descrita no capítulo quatro.

## 3.3 O Browser de Banco de Dados "nas Interfaces Visuais"

O browser de banco de dados tem a finalidade de iniciar uma sessão de trabalho, ou seja, o usuário seleciona ou abre um banco de dados para depois navegar pelo esquema, classes ou objetos. A idéia básica do browser de banco de dados é mostrar ao usuário os bancos de dados disponíveis, ou através de uma lista ou através de uma representação gráfica (ícones).

O browser de banco de dados pode ser comparado em GOODIES com a *Janela de Banco de Dados (BD)*, onde é apresentado o *rol* de classes que compõem o esquema do banco de dados, que é selecionado previamente através da *Janela de Diretórios*. A figura 3.1 mostra a *Janela de BD* de GOODIES, contendo as classes do banco de dados *Sistema*.

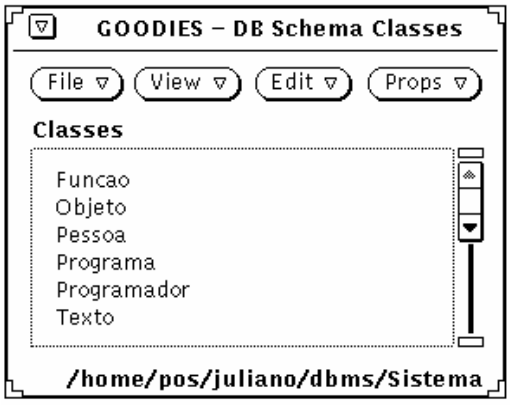



FIGURA 3.1- A *Janela de Banco de Dados* da interface GOODIES [OLI 93]

Já a interface gráfica do ODE, OdeView, apresenta o browser de banco de dados como uma janela inicial que contém ícones, os quais representam bancos de dados gerenciados pelo SGBDOO ODE. A escolha de um banco de dados abre uma janela que contém a descrição gráfica da hierarquia de classes. A figura 3.2 mostra a janela inicial de OdeView.

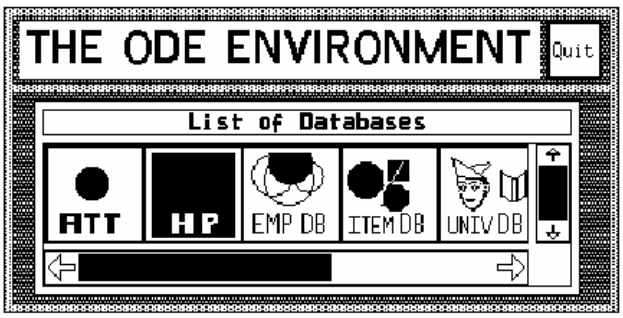



FIGURA 3.2- A janela inicial de OdeView [AGR 90]

## 3.4 O Browser de Esquema "nas Interfaces Visuais"

O Browser de Esquema permite ao usuário visualizar ou atualizar a hierarquia de classes (superclasses, subclasses, relacionamentos) do banco de dados, escolhido através do browser de banco de dados. A idéia básica desse browser é permitir ao usuário visualizar e manipular o esquema de um banco de dados graficamente. O browser de esquema, por ser considerado visual, deve representar o esquema do banco de dados (a hierarquia de classes), seguindo algum formalismo.

Na interface *ObjectStore Inspector* [VIV 96], o usuário pode escolher em que formalismo deseja visualizar o esquema do banco de dados (a representação gráfica). Pode-se visualizar o esquema em uma dessas abstrações: (1) OMT-Rumbaugh [RUM 91]; (2) Booch [BOO 91]; (3) Coad-Yourdan [COA 91]; (4) Statecharts [HAR 87]. A figura 3.3 mostra a representação gráfica de um esquema, através da interface *ObjectStore Inspector*, utilizando a abstração de Booch.

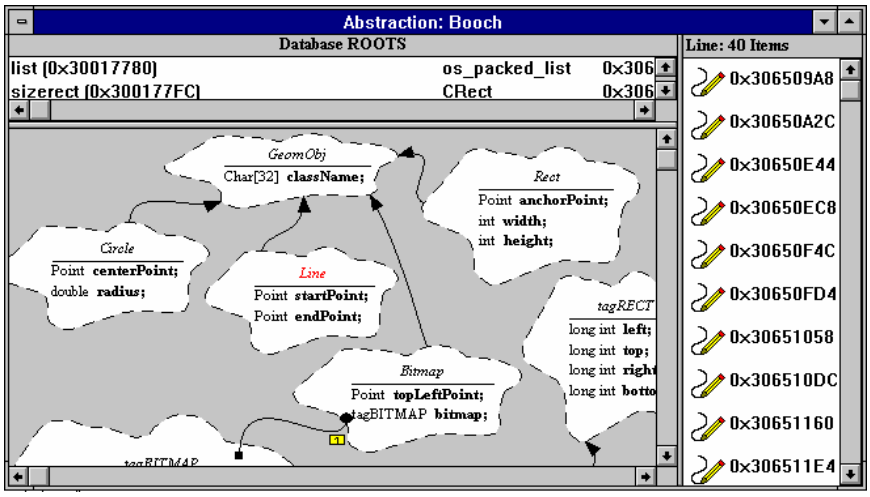



FIGURA 3.3- *ObjectStore Inspector*, exibindo um esquema graficamente na abstração de Booch [VIV 96]

O SGDOO $O_2$ [$O_2$ 95] possui um ambiente de programação, denominado $O_2$Tools, para o desenvolvimento de aplicações. Esse ambiente permite, entre outras funcionalidades, a manipulação da hierarquia de classes. $O_2$Tools apresenta ferramentas para simplificar o trabalho do programador: (1) $O_2$ *browser*; (2) $O_2$ *shell*; (3) *Workspace*; (4) *Debugger*. O *browser* é o ponto inicial de qualquer sessão de programação no $O_2$. Desde aplicações, classes, programas e métodos são representados como objetos. Esses objetos são acessados pelo programador através desse *browser*. A figura 3.4 mostra a representação gráfica de uma *Hierarquia de Classes* em $O_2$.

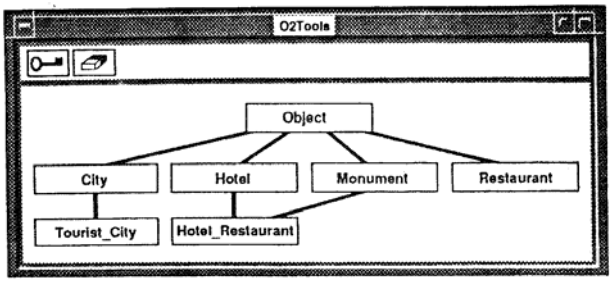



FIGURA 3.4- A representação gráfica de uma *Hierarquia de Classes* através do *Browser* de $O_2$Tools [DEU 91]

## 3.5 O Browser de Classe "nas Interfaces Visuais"

O Browser de Classe permite ao usuário visualizar ou atualizar a definição de uma classe, selecionada através do browser de esquema. A idéia básica desse browser é apresentar a visão de uma única classe por vez, através de uma descrição textual dos seus atributos e métodos. Nesse browser tem-se uma visão mais regional do esquema, pois são apresentadas apenas a classe e suas relações (superclasses, subclasses e classes agregadas).

A figura 3.5 mostra a *Janela de Classe* de GOODIES, contendo a descrição da classe Programa, apresentada na lista de classes da figura 3.1.

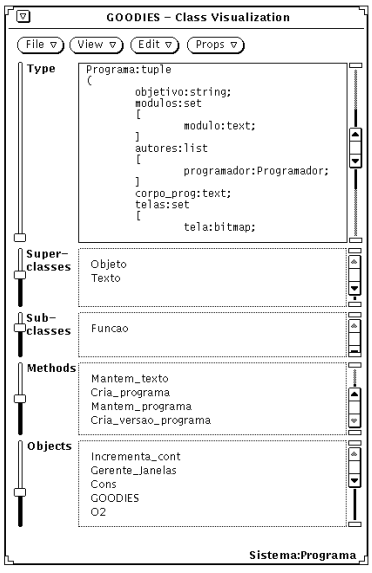



FIGURA 3.5- A *Janela de Classe* da interface GOODIES [OLI 93]

O browser de classe pode ser comparado em GOODIES com a *Janela de Classe* (ver figura 3.5), onde é apresentada a definição da classe, através de uma descrição textual do seu tipo. Também consta nessa janela a listagem das suas superclasses, subclasses, métodos e objetos.

Já na interface gráfica OdeView a definição de uma classe é apresentada na *Janela de Informações da Classe*, que é dividida em três *viewports*: (1) a primeira mostra as superclasses; (2) a segunda as subclasses; e (3) a terceira contém os meta-dados associados à classe, como mostra a figura 3.6. OdeView também permite visualizar a definição da classe, através de uma descrição textual. O usuário pode navegar no esquema, selecionando outro nó do grafo de hierarquia, ou selecionando uma classe nas *viewports* de superclasse ou subclasse.

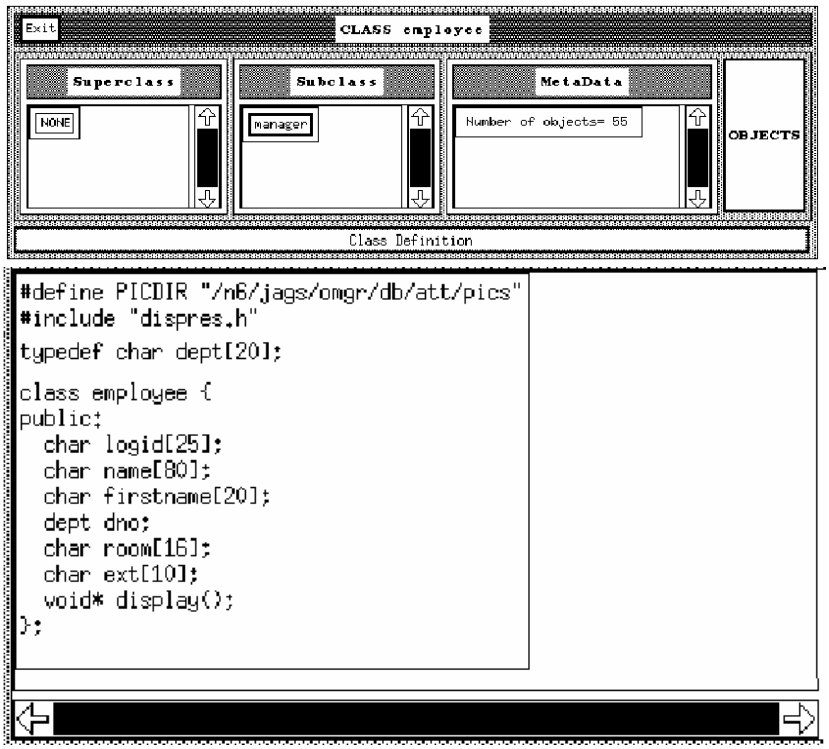



FIGURA 3.6- A definição de uma classe em OdeView [AGR 90]

## 3.6 O Browser de Objeto “nas Interfaces Visuais”

O Browser de Objeto permite ao usuário visualizar ou atualizar os objetos de uma classe, versionados (versões) ou não-versionados [GOL 95]. A idéia básica desse browser é permitir ao usuário visualizar ou atualizar os atributos de um objeto em seus diferentes níveis na hierarquia de classes, e as versões desse objeto, que são apresentadas através de uma representação gráfica (grafo de versões [GOL 95]).

Na interface *ObjectStore Inspector* o usuário pode escolher em que representação gráfica deseja visualizar as instâncias (objetos) do banco de dados. *ObjectStore Inspector* mostra as instâncias através de: endereço, valor e ícone, ou ambos.

A figura 3.7 mostra a representação gráfica das instâncias da classe *Vehicle*, do banco de dados *ROOTS*, através da interface *ObjectStore Inspector*, utilizando a representação de: endereço, valor e ícone.

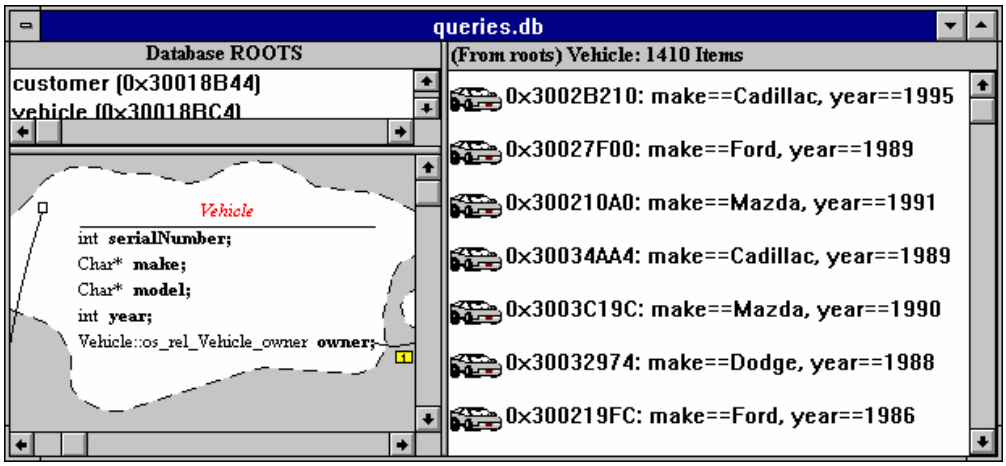



FIGURA 3.7- *ObjectStore Inspector*, exibindo graficamente instâncias de uma classe

O browser de objeto pode ser comparado em GOODIES com a *Janela de Objetos*, onde são apresentados os valores dos atributos que compõem a instância da classe. Os atributos são divididos nos seguintes grupos: Simples, Compostos, Bitmaps, Sounds, Listas, Tuplas e Subobjetos. A figura 3.8 mostra a *Janela de Objetos* de GOODIES, contendo a descrição de uma instância (objeto) da classe *Programa*.

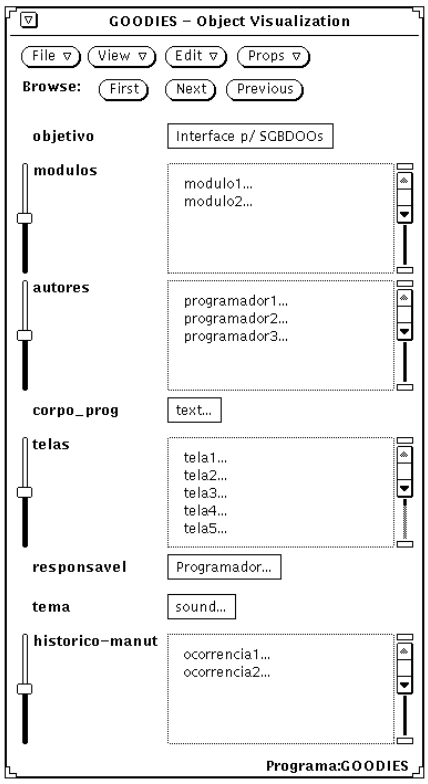



FIGURA 3.8- A *Janela de Objetos* da interface GOODIES [OLI 93]

Na interface OdeView, é possível visualizar e atualizar, através da navegação, além dos objetos simples também os objetos complexos. A janela de informações de classes tem um botão *objects* que permite aos usuários navegarem sobre os dados. Esse botão causa a abertura da *Janela de Apresentação de Objetos*, onde são apresentados os objetos pertencentes à classe, em um ou mais formatos, dependendo da semântica dos métodos. A figura 3.9 mostra a Janela de Apresentação de Objetos de OdeView.

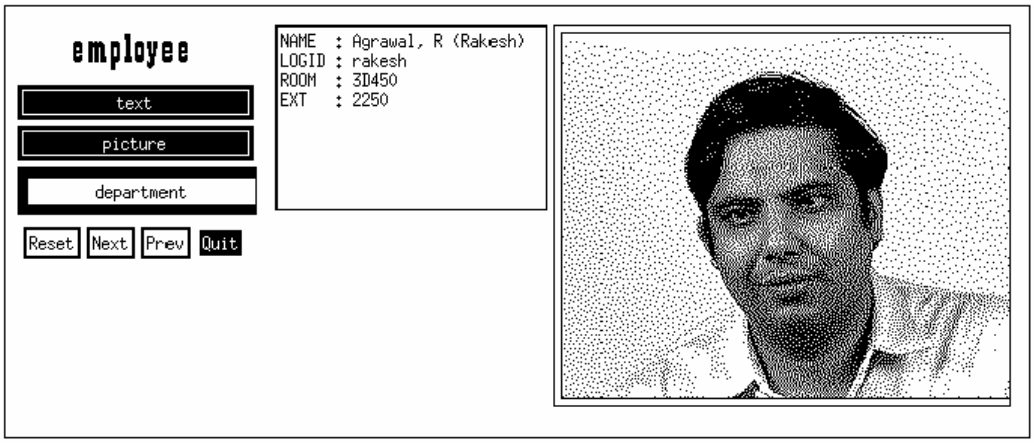



FIGURA 3.9- A *Janela de Apresentação de Objetos* de OdeView [AGR 90]

$O_2$ apresenta um gerador de interface, denominado de $O_2$Look, que suporta a exibição e a manipulação de objetos grandes, objetos complexos e objetos multimídia [DEU 91]. A figura 3.10 mostra uma janela de $O_2$, exibindo um objeto com atributos complexos e do tipo Imagem (*Bitmap*).

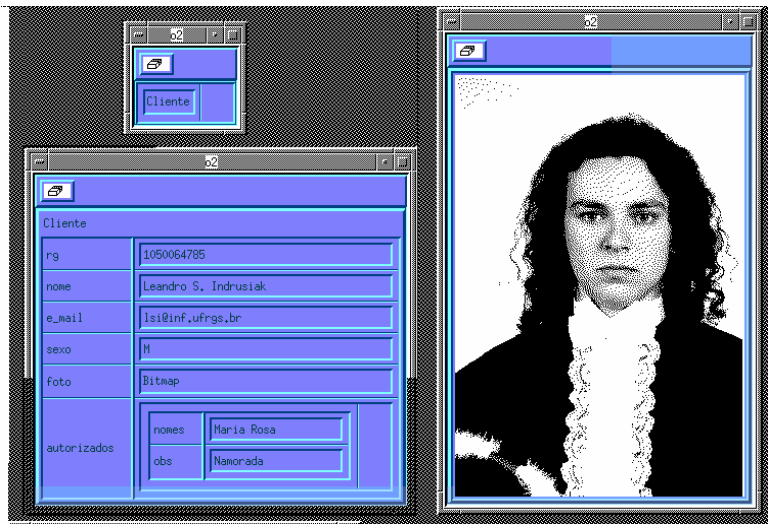



FIGURA 3.10- A representação gráfica de um objeto em $O_2$ [SIL 96]

## 3.7 O Browser de Consulta “nas Interfaces Visuais”

O Browser de Consulta permite ao usuário formular consultas gráficas interativamente, através de filtros informados diretamente nas caixas de edição dos atributos, restringindo a navegação para um subconjunto de objetos ou versões.

Uma interface que representa bem um *browser* para consulta é a interface PESTO [CAR 96, CAR 95]. PESTO permite a navegação e a consulta aos objetos e aos relacionamentos entre os objetos. Os usuários formulam suas consultas a objetos complexos através de um paradigma de consulta integrado que apresenta consultas como extensão natural de navegação. PESTO foi projetado para ser portável para qualquer sistema de banco de dados de objetos que suporte uma linguagem de consulta de alto nível. PESTO é extensível, fornecendo opções para a formação de predicados especializados e ferramentas de exibição de objetos para novos tipos de dados (p.e., imagens ou texto).

Segundo Carey, PESTO se diferencia de outros sistemas analisados em [CAR 96] por permitir aos usuários restringirem diretamente a exibição dos dados através de filtros: (1) na estrutura do browser, (2) em qualquer nível do conjunto, e (3) com predicados complexos (incluindo ligações implícitas e explícitas). A figura 3.11 mostra um exemplo do resultado de uma consulta em PESTO, permitindo navegação sincronizada.

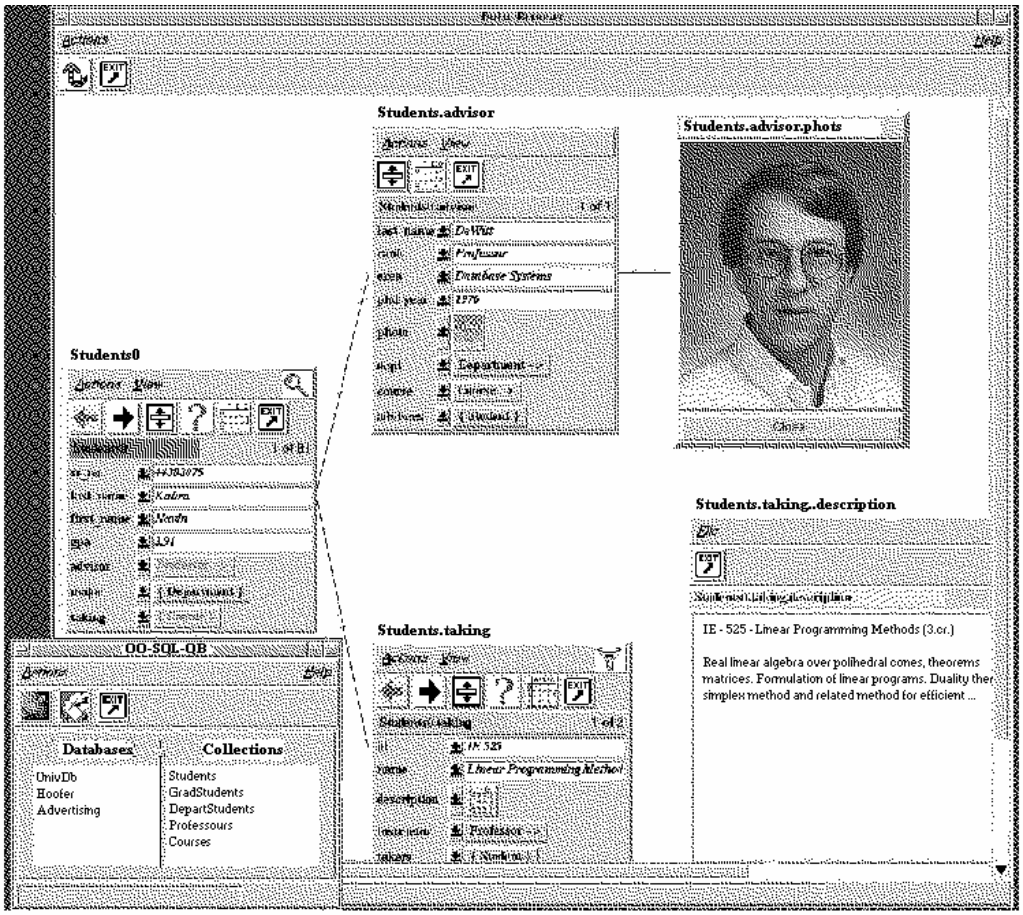



FIGURA 3.11- Consulta e navegação sincronizada com PESTO [CAR 96]

Na janela “OO-SQL/QB”, da figura 3.11, aparece uma lista dos bancos de dados disponíveis. No exemplo foi escolhido para explorar o BD *UnivDB*, onde aparece uma lista com as suas classes. Foi selecionada a classe *Students*, servindo de ponteiro para a navegação (como mostra a janela chamada ‘Students0’, na figura 3.11). O usuário pode navegar pelos relacionamentos que existem entre os objetos e seus subobjetos, assim como navegar/consultar os outros objetos das classes do banco de dados. Uma descrição detalhada do funcionamento da figura 3.11. pode ser encontrada em [CAR 96].

A interface GOODIES apresenta facilidades de consulta que tornam o seu mecanismo de navegação mais poderoso e simples. Oliveira ressalta a terminologia adotada, diferenciando navegação de consulta.

- *Navegação:* é o processo de visualização seqüencial de informações de um determinado tipo;

- *Consulta:* é o processo de seleção e restrição das informações, de modo que apenas as informações explicitamente solicitadas sejam recuperadas e apresentadas ao usuário.

As principais facilidades de consulta apresentadas pela interface GOODIES são: (1) os predicados; e (2) a navegação sincronizada.

- *Predicados:* um predicado é formado por três elementos: (1) *atributo:* um atributo do objeto representado na janela de objeto para a qual foi solicitada a criação da janela de predicados; (2) *operador:* um operador de comparação (=,<,>, $\leq$, $\geq$, $\neq$) ou de conjunto ($\supset$, $\subset$); e (3) *referencial:* um valor ou atributo de objeto representado na tela, cujo tipo seja igual ao do atributo correspondente ao primeiro elemento do predicado.

- *Sincronismo:* uma janela de objeto pode conter referências a outros objetos (subobjetos), estabelecendo uma ligação de sincronismo entre a janela do objeto complexo e a janela criada para o subobjeto. Cada referência a um sub-objeto pode possuir diversas janelas associadas, formando uma *árvore de sincronização*, como mostra a figura 3.12. Uma ligação de sincronismo não é permitida entre dois objetos quaisquer. Uma janela de objeto só pode ser *pai* de outra janela na árvore de sincronização, se o objeto representado naquela janela contém algum atributo que referencia esse objeto.

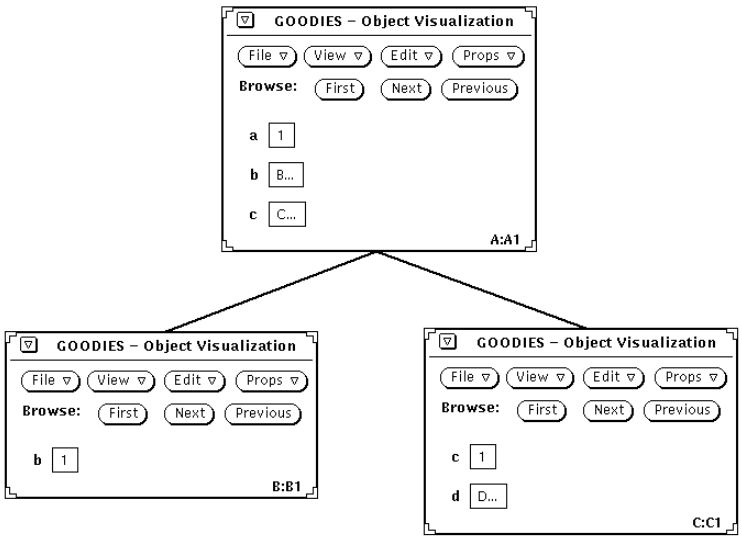



FIGURA 3.12- Árvore de Sincronização da Interface GOODIES

Na interface *ObjectStore Inspector*, o usuário pode definir consultas complexas, usando o mecanismo "*Click and Query*". A classe envolvida na definição da consulta é destacada no esquema. O usuário pode atribuir restrições (predicados) de valores nos próprios campos, usando mecanismos visuais. A expressão de consulta equivalente é automaticamente gerada pela Interface *ObjectStore Inspector*. As consultas também podem ser formuladas diretamente, utilizando a interface de expressões de consulta do *ObjectStore*.

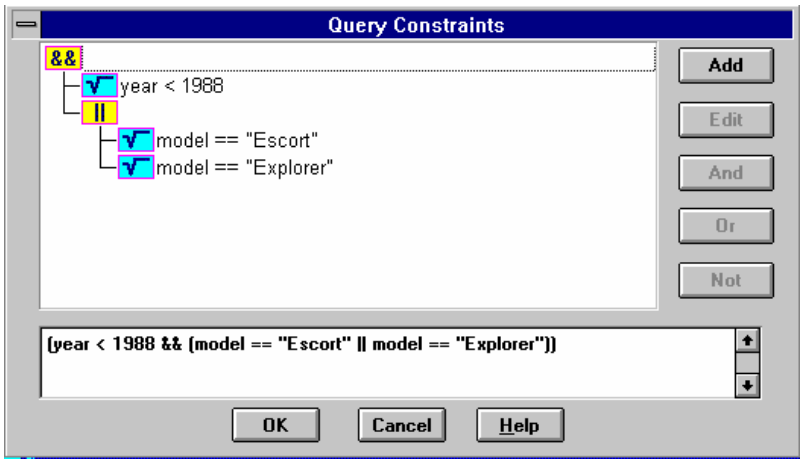



FIGURA 3.13- Expressões em *ObjectStore Inspector*

Observou-se a importância de se analisarem características de outras interfaces para modelos de dados orientados a objetos, sejam experimentais ou comerciais, durante o projeto de uma interface. Pois, implementações e enfoques diferenciados aumentam o leque de possibilidades de se projetarem boas funcionalidades na interface, entre outros aspectos.

# 4 O Projeto da interface

Este capítulo descreve o projeto de uma interface visual para modelos de bancos de dados orientados a objetos, com suporte para versões.

A proposta inicial direcionou o projeto de *layout* da interface para o desenvolvimento de uma "Windows® like". Esse padrão de interface pode ser facilmente observado em inúmeros sistemas atualmente disponíveis para o ambiente *Windows* (editores de texto, planilhas eletrônicas). O projeto de *layout* da interface seguiu como recomendação a guia de estilo [MS 97], da Microsoft® Corporation, para o Sistema Operacional Microsoft® Windows® 95.

A interface é basicamente composta por uma tela principal e janelas (formulários) de *browsers*, onde podem ser visualizados o banco de dados, o esquema, as classes, os objetos, as versões dos objetos, entre outras informações.

A FIGURA 4.1, conforme [MS 97], apresenta o *layout* da janela principal da interface, que é composta por:

(1) *Barra de Título (Title Bar):* também referenciada como barra de capítulo (*caption bar*), identifica qual a janela que está sendo visualizada;

(2) *Ícone da Barra de Título (Title Bar Icons)*: aparece no canto superior esquerdo da janela principal e representa o objeto que está sendo visualizado;

(3) *Título Textual (Title Text)*: identifica o nome do objeto que está sendo visualizado na janela;

(4) *Botões da Barra de Título (Title Bar Buttons)*: botões, como teclas de atalhos (*shortcut*), com comandos específicos para a janela (fechar, minimizar, maximizar e restaurar);

(5) *Barra de Menu (Menu Bar)*: inclui um conjunto de entradas chamadas de *títulos de menu*. Cada título de menu fornece o acesso para um *menu drop-down*, composto por uma coleção de itens de menu;

(6) *Barra de Ferramenta (Toolbars):* é um painel que contém um conjunto de controles (botões), projetados para fornecer um acesso rápido para comandos específicos ou opções;

(7) *Barra de Status (Status Bars)*: mostra informações sobre o estado corrente do que está sendo visualizado na janela ou outras informações do contexto, como o *status* do teclado.

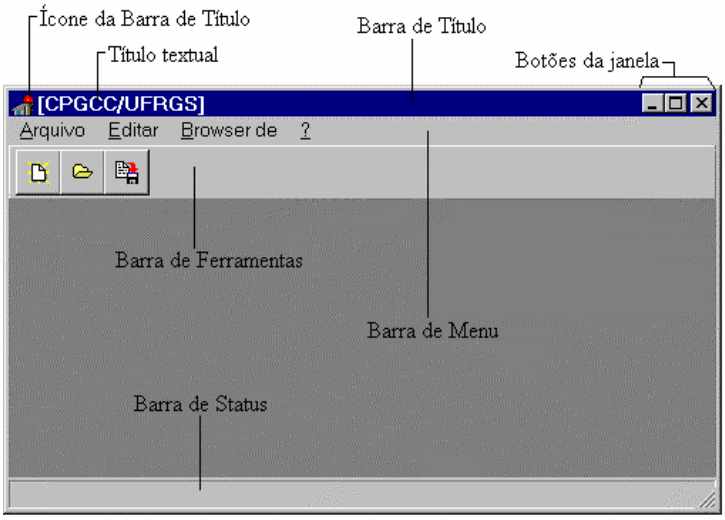



FIGURA 4.1- O *layout* da janela principal da interface

## 4.1 A Hierarquia de acesso aos browsers

A interface é composta por um conjunto de janelas (formulários) de *browsers* que representam os diversos níveis de visualização das informações de um banco de dados de objetos. Os browsers que compõem a interface são:

(1) browser de banco de dados;
(2) browser de esquema;
(3) browser de classe;
(4) browser de objeto (suporte para versões);
(5) browser de consulta.

A figura 4.2 mostra a hierarquia dos browsers e os caminhos (fluxo de acesso) que podem ser percorridos para ativar um browser. Os demais browsers só podem ser ativados depois que um banco de dados for aberto através do browser de banco de dados.

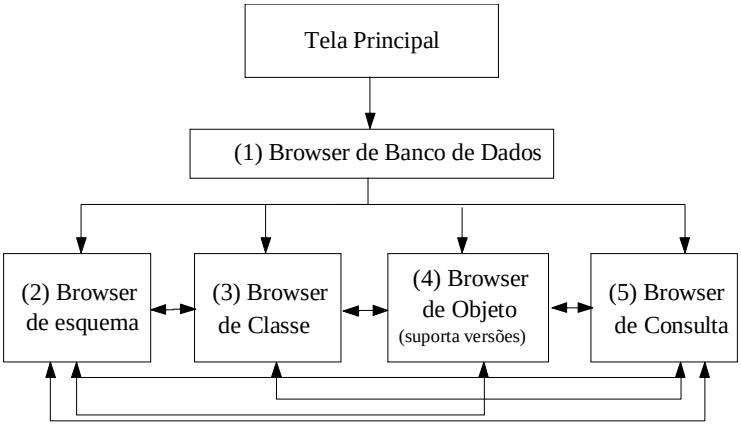



FIGURA 4.2- A hierarquia de acesso aos browsers

## 4.2 Os Modos de Trabalho da Interface

Um aspecto, a ser destacado, é o modo de trabalho da interface, pois a interface se propõe a ser uma interface "genérica" e "específica". No caso, a interface será considerada específica por almejar manipular bases de dados do banco de dados orientados a objetos $O_2$. Exemplos de interfaces específicas: *ObjectStore Inspector*, interface específica para o BDOO ObjectStore Object [VIV 96], OdeView, a interface visual específica para o BDOO ODE [AGR 90], a interface gráfica específica para o BDOO $O_2$ [$O_2$ 95]. Também será considerada "genérica", por almejar manipular "qualquer" BDOO que mapeie suas bases de dados para o padrão ODMG-93 [CAT 96]. Exemplos de interfaces genéricas: GOODIES [OLI 93, OLI 94] e PESTO [CAR 96, CAR 95]

E os browsers, seguindo esse propósito, estarão em um dos seguintes modos de trabalho: (1) modo de atualização; ou (2) modo de visualização.

- *Modo de Atualização:* a idéia básica nesse modo de trabalho é permitir ao usuário manipular um banco de dados do BDOO $O_2$ diretamente (seu esquema, classes, objetos e versões), através da interface proposta. A base de dados poderá ser criada tanto pela interface como pelo próprio SGBDOO $O_2$. É importante salientar que esse banco de dados pode ser ***visualizado*** e, principalmente, ***atualizado***.

- *Modo de Visualização:* A idéia básica nesse modo de trabalho é permitir ao usuário visualizar o esquema e os objetos de um banco de dados de "outro SGBDOO", além do SGBDOO $O_2$. Esse banco de dados só pode ser ***visualizado***. Para a interface conseguir visualizar o banco de dados, ele deve estar descrito segundo o padrão ODMG-93 [CAT 96].

## 4.3 O Browser de Banco de Dados

No browser de banco de dados da interface optou-se por representar graficamente, através de ícones, os bancos de dados disponíveis, para que o usuário selecione um e inicie uma sessão de trabalho. A figura 4.3 mostra o *layout* do browser de banco de dados projetado para a interface.

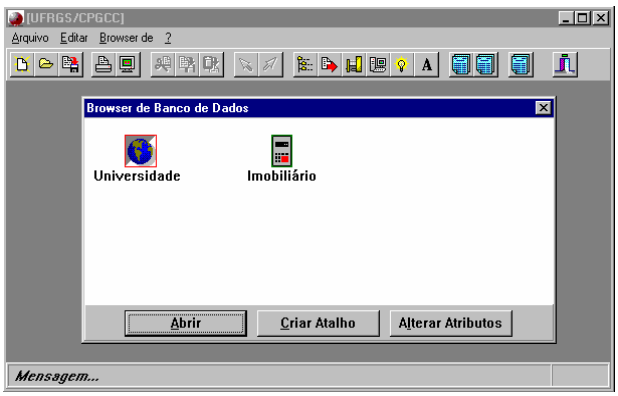



FIGURA 4.3- O *layout* Browser de Banco de dados

O Browser de Banco de Dados, como mostra a figura 4.3, por *default*, é ativado, quando a interface é executada, pois esse browser tem a finalidade de iniciar uma sessão de trabalho. Pode-se, também, ativar esse browser através da opção *Browser de Banco de Dados*, na barra de menu ou na barra de ferramentas.

O processo de navegação na interface se estabelece, quando um banco de dados de objetos é aberto ou quando é criado um novo banco de dados. As características presentes no browser de banco de dados, disponíveis aos usuários, são as seguintes:

- *criar um novo BDO $O_2$:* essa opção está disponível através do botão ***Novo,*** na barra de ferramentas. Após a confirmação, o *Browser de Esquema* é ativado, para que possa ser projetado o esquema. A figura 4.4 mostra a janela para a criação de um novo banco de dados.

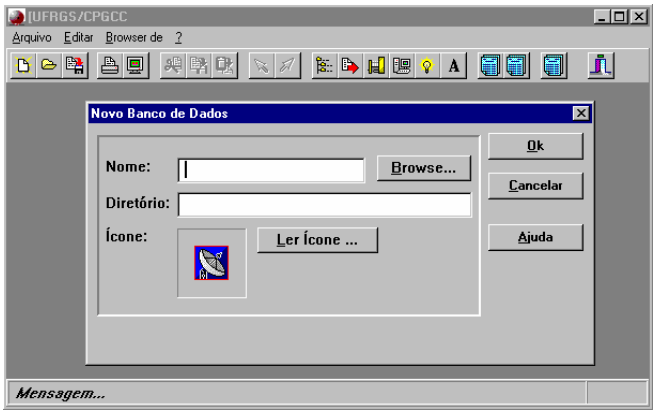



FIGURA 4.4- Janela para criação de um novo banco de dados

- *estabelecer um atalho:* para um banco de dados que já existe e que ainda não foi lido (manipulado) pela interface. Esse banco de dados pode ser do SGBDOO $O_2$ (visualizado e atualizado) ou de outro SGBDOO (visualizado). O banco de dados terá um ícone, escolhido pelo usuário, para o identificar. Essa opção está disponível através do botão ***Criar Atalho***, na barra de botões específicos. A figura 4.5 mostra a janela para se estabelecer um atalho para um banco de dados.

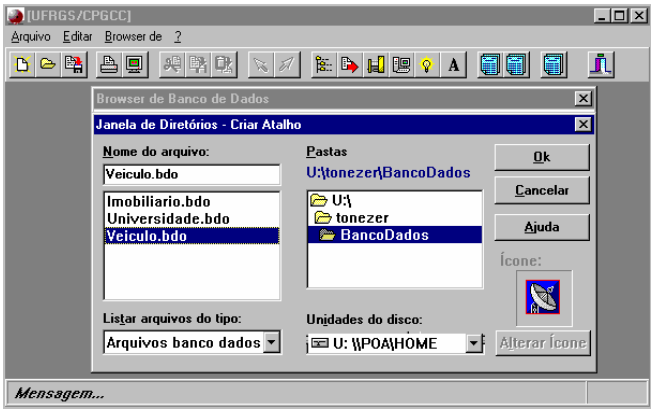



FIGURA 4.5- Estabelecer um Atalho para um banco de dados

- *abrir um banco de dados:* que já possua um ícone de atalho. A abertura é através da seleção do ícone, seguido do clique no botão *Abrir*, opção disponível na barra de botões específicos. Essa opção ativa o *Browser de Esquema*, para ser visualizado;

- *alterar atributos do BDO:* (1) nome do BDO; (2) diretório onde está o BDO; (3) ícone de atalho que identifica o BDO. Essa opção está disponível através do botão *Alterar Atributos*, na barra de botões específicos. A janela para alterar os atributos de um BDO é semelhante à da figura 4.4.

## 4.4 O Browser de Esquema

No browser de esquema da interface optou-se por representar graficamente a hierarquia de classes, utilizando o formalismo da UML (**U**nified **M**odeling **L**anguage) [RAT 97a], por ser a UML uma unificação de várias metodologias. Essa metodologia de modelagem orientada a objetos é sucessora das linguagens de modelagem encontradas em Booch [BOO 91], Rumbaugh-OMT [RUM 91], Jacobson (OOSE: Object-Oriented Software Engineering), e outros métodos. Uma implementação dessa linguagem de modelagem pode ser encontrada na ferramenta CASE *Rational Rose* [RAT 97b].

A figura 4.6 mostra o *layout* do Browser de Esquema, com a representação gráfica de uma hierarquia de classes (esquema).

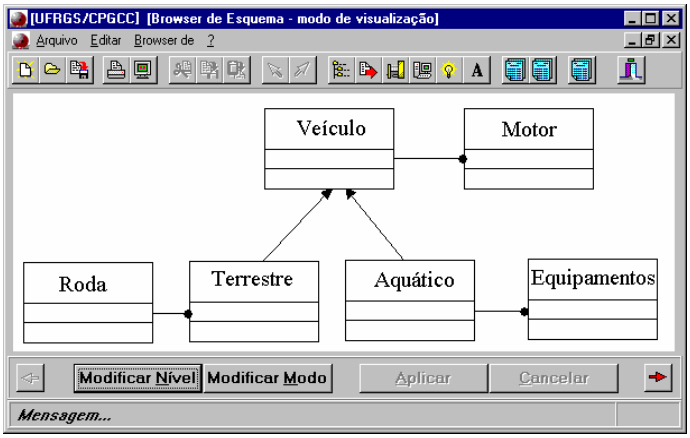



FIGURA 4.6- O *layout* do Browser de Esquema

Como a interface se propõe a ser uma interface genérica e específica (ver item 4.2), o Browser de Esquema também deverá estar em um dos modos de trabalho: atualização ou visualização.

- *modo de atualização:* o usuário poderá atualizar as informações no nível de esquema ou, no caso de um novo BDO, projetar o seu esquema. Cabe ressaltar que o banco de dados a ser atualizado é do SGBDOO $O_2$.

- *modo de visualização:* esse modo é utilizado apenas para visualizar o esquema. A interface estará nesse modo, quando o banco de dados aberto for de outro SGBDOO que não o $O_2$.

Na interface optou-se por representar o esquema do banco de dados (as classes) em dois níveis de detalhamento das informações:

- *nível 1:* no primeiro nível tem-se uma visão mais global do esquema, com apenas o nome da classe e os relacionamentos entre as classes (classe - superclasse - classe agregada), como a representação gráfica do $O_2$. A figura 4.7 mostra a notação gráfica utilizada para representar uma classe no nível 1 de detalhamento.

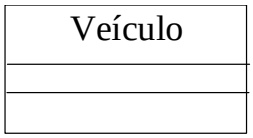



FIGURA 4.7- Representação gráfica de uma classe no nível 1

- *nível 2:* no segundo nível tem-se uma visão mais regional do esquema, pois, na maioria dos casos, não será possível ver todo o esquema, necessitando de *scroll*[1]. Nesse nível serão visualizados, além do nome da classe e seus relacionamentos, os atributos da classe, como a representação gráfica de *ObjectStore Inspector*. A figura 4.8 mostra a notação gráfica utilizada para representar uma classe no nível 2 de detalhamento.

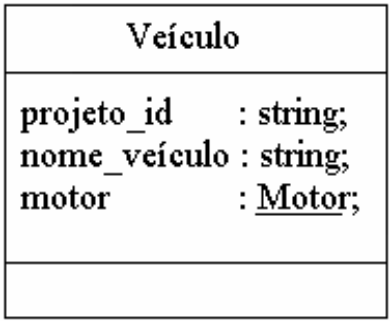



FIGURA 4.8- Representação gráfica de uma classe no nível 2

### 4.4.1 A Barra de Status do Browser de Esquema

Conforme [MS 97], a barra de status é uma área especial da janela, tipicamente no rodapé, que mostra informações do estado corrente ou do contexto. Normalmente, a barra de status inclui apenas informações não-interativas ou somente de leitura. Mas,

[1] Conforme [MS 97], quando a informação visualizada na janela exceder o tamanho dessa janela, a janela deve suportar rolagem (*scrolling*). Scroll fica disponível através de uma *Barra de Scroll*, na horizontal ou vertical.

como a barra de ferramenta, a barra de status pode conter botões de comando. As opções disponíveis (botões) na barra de status da janela do browser de esquema são:

-  *Seta para a Esquerda:*
  - ⇒ não está disponível para esse browser;
  - ⇒ sua finalidade é ativar o browser anterior. Nesse caso, o browser ativo é o de Esquema e o seu anterior é o browser de Banco de Dados.

-  *Seta para a Direita:*
  - ⇒ está disponível para esse browser;
  - ⇒ sua finalidade é ativar o browser seguinte. Nesse caso, o browser posterior é o *Browser de Classes*.

- *Modificar **N**ível:*
  - ⇒ modifica o nível de visualização das informações, conforme o nível ativo. Como exemplo, ao pressionar esse botão, na figura 4.6 o *browser* de esquema exibirá o esquema com mais informações (nível 2), utilizando a representação gráfica, apresentada na figura 4.8.

- *Modificar **M**odo:* modifica o modo de trabalho, conforme o tipo de banco de dados (ver item 4.2). Se o banco de dados for apenas para visualização, esse botão estará desabilitado. Se o banco de dados for do SGBDOO $O_2$, ou seja, permita, além da visualização, a atualização do esquema, o *browser* tornará visível uma barra de atalho com as opções para se projetar o esquema (ver item 4.4.2).

- ***A**plicar:*
  - ⇒ disponível apenas no modo de Atualização;
  - ⇒ atualiza as últimas alterações feitas no esquema do BDO ativo;

- ***C**ancelar:*
  - ⇒ disponível apenas no modo de Atualização nos dois níveis de detalhamento;
  - ⇒ cancela as últimas alterações feitas no esquema do BDO ativo no browser, voltando ao estágio inicial da última atualização.

### 4.4.2 A Caixa de Ferramentas para o projeto do Esquema

O Browser de Esquema, no modo atualização, apresenta uma caixa de ferramentas com opções disponíveis para o projeto do esquema. Essas opções são representadas graficamente por ícones. Essas funcionalidades foram baseadas no *Diagrama de Classes*[1], da ferramenta *Rational Rose* [RAT 97b].

A figura 4.9 mostra a caixa de ferramentas com os ícones que representam as opções para o projeto do esquema.

[1] O *Diagrama de Classes* é apenas um dos diagramas disponíveis na ferramenta Rational Rose. O nome mais correto para esse diagrama seria *Diagrama Estático Estruturado*, mas *Diagrama de Classe* soa melhor [RAT 97b].

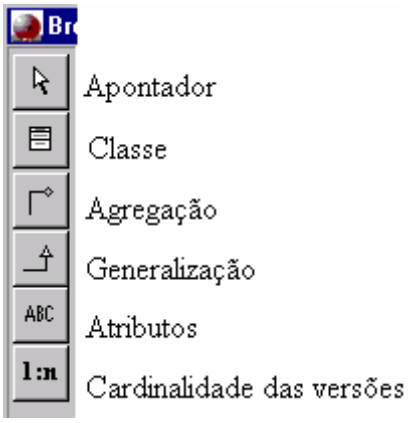



FIGURA 4.9- A Caixa de Ferramenta para o projeto do esquema

A caixa de ferramentas é um painel que contém um conjunto de controles (botões), projetados para fornecer um acesso rápido a opções específicas [MS 97]. A caixa de ferramenta pode ser considerada como uma especialização da barra de ferramenta. As opções disponíveis (botões) na caixa de ferramenta da janela do browser de esquema são:

- *Apontador* : essa opção é utilizada para desmarcar uma opção selecionada;
- *Classe*: essa opção é utilizada para incluir uma nova classe ao esquema;
- *Agregação*: essa opção é utilizada para criar uma agregação entre classes;
- *Generalização*: essa opção é utilizada para especializar uma classe (superclasse ← classe ← subclasse);
- *Atributos*: essa opção é utilizada para inserir os atributos de uma classe. Essa opção deve ser selecionada e arrastada até a classe em que se desejam incluir os atributos. Ao ser selecionada, ativa automaticamente o *nível* 2 de detalhamento. Os atributos da classe também podem ser inseridos através do Browser de Classe (descrito no item 4.5), que é o *browser* recomendado para a realização dessa tarefa. Essa opção foi mantida aqui para seguir a padronização proposta para o nível 2 (ver figura 4.8.);
- *Cardinalidade das versões*: essa opção é utilizada para determinar a cardinalidade das correspondências (mapeamentos) entre as versões de um objeto em uma *classe* (descendente, extensão) e versões de seu ascendente (protótipo) na *superclasse*. No modelo de versões [GOL 95], as correspondências são classificadas em: *1:1*, *n:1*, *1:n* e *n:m*.

*Na correspondência 1:1, cada versão na subclasse deverá corresponder exatamente a uma versão na superclasse, isto é, quando uma versão for criada na subclasse, deverá ser ligada a uma versão na superclasse. Cada versão na superclasse poderá corresponder a, no máximo, uma versão na subclasse, isto é, pode ser criada uma versão na superclasse, sem descendente correspondente.*

*Na correspondência n:1, várias versões na superclasse poderão corresponder a uma versão na subclasse e só uma versão na subclasse poderá corresponder a cada versão na superclasse.*

*Na correspondência 1:n, uma versão na* subclasse *pode corresponder a várias versões na* superclasse *e várias versões na* superclasse *poderão corresponder a uma versão na* subclasse*.*

*Na correspondência n:m, várias versões na subclasse poderão estar relacionadas com uma versão na superclasse e a cada versão na subclasse poderão corresponder várias versões na superclasse.*

As interfaces analisadas: [VIV 96], [AGR 90], [OLI 93, OLI 94], [CAR 96, CAR 95], para sistemas de banco de dados orientados a objetos, não contemplam as correspondências entre as versões nos vários níveis da hierarquia de herança.

### 4.4.3 O Projeto de um esquema

Para ficar mais claro o entendimento do browser de esquema será mostrado um pequeno estudo de caso para o projeto de um esquema exemplo. O esquema exemplo é composto por seis classes: *Veículo, Terrestre*, *Aquático, Motor, Roda* e *Equipamentos*; e um método da classe *Veículo*: *IncluiVeículo*.

- *Esquema exemplo: "Projeto de Veículos":*

⇒ *Classe* **Veículo {**
projeto_id : string,
nome_veículo : string,
motor : **Motor**; // agregação **}**

⇒ *Classe* **Terrestre** *"herança de"* **Veículo {**
roda : **Roda** // agregação,
nro_eixos : integer; **}**

⇒ *Classe* **Aquático** *"herança de"* **Veículo {**
calado : real,
equipamentos : lista(**Equipamentos**); // agregação **}**

⇒ *Classe* **Motor** *"componente de"* **Veículo {**
potência : integer,
combustível : string; **}**

⇒ *Classe* **Roda** *"componente de"* **Terrestre {**
diâmetro : real,
largura : real,
estilo : string,
material : string; **}**

⇒ *Classe* **Equipamentos** *"componente de"* **Aquático {**
nome : string,
descrição : lista(string); **}**

// **Métodos**
⇒ *Método* IncluiVeículo em *Classe* Veículo

```
program body Veículos in application ProjetoVeículos
{
        o2 Veículo V = new Veículo;

        if (V -> edit == SAVE)
        {
                Veículos += set(V);
                V = new Veículo;
        }
}
```

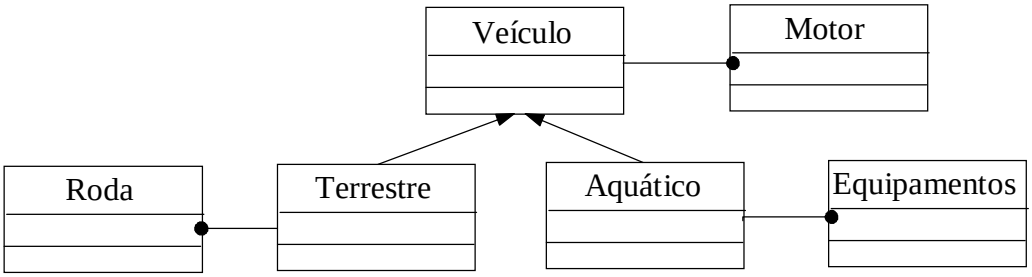



FIGURA 4.10- Representação gráfica do Esquema-exemplo

- *Etapas sugeridas para o projeto do esquema exemplo:*

⇒ ***1ª Etapa:*** incluir um novo BDO: ao confirmar a inclusão do BDO, o Browser de Esquema vai ser ativado no *Modo de Atualização*, nível 1 de detalhes;

⇒ ***2ª etapa:*** incluir as classes: selecionar o ícone de nova classe  e confirmar a opção na tela. Um retângulo vai ser visualizado (que representa uma classe) e ficará um cursor piscando, esperando que seja informado o nome da classe, como mostra a figura 4.11.


FIGURA 4.11- Incluindo a primeira classe ao esquema

- incluir as demais classes do esquema;

⇒ ***3ª etapa:*** estabelecer os relacionamentos entre as classes: generalizações e agregações;

- *como estabelecer uma agregação:* um exemplo de agregação entre duas classes do esquema é entre as classes *Veículo* e *Motor* (é componente de Veículo). Os passos são os seguintes:

- selecionar o ícone da agregação  ;
- selecionar a classe origem da agregação (que, no exemplo, é a classe *Veículo*) e confirmar com um clique;
- selecionar a classe destino da agregação (que, no exemplo, é a classe *Motor*) e confirmar com um clique;

A figura 4.12 mostra, graficamente, a agregação estabelecida entre as classes *Veículo* e *Motor*.

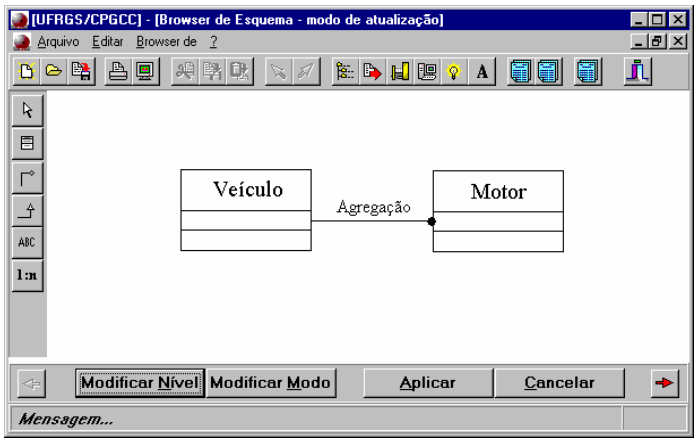



FIGURA 4.12- Estabelecendo uma Agregação

- *como estabelecer uma generalização:* um exemplo de generalização entre duas classes do esquema é entre as classes *Terrestre* (é uma subclasse de) e *Veículo*. Os passos são os seguintes:

- selecionar o ícone de generalização  ;
- selecionar a subclasse que herdará os atributos (que, no exemplo, é a classe *Terrestre*) e confirmar com um clique;
- selecionar a superclasse (que, no exemplo, é a classe *Veículo*);

A figura 4.13 mostra, graficamente, a generalização estabelecida entre as classes *Veículo* e *Terrestre*. Na figura 4.13, é visualizado um esquema com poucas classes, mas, no caso de um esquema com mais classes, a interface disponibilizará uma barra de scroll, na posição vertical e/ou horizontal.

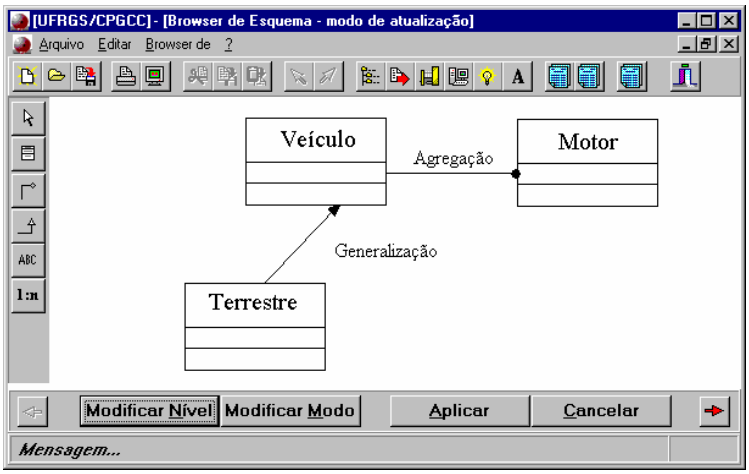



FIGURA 4.13- Estabelecendo uma Generalização

***obs.:*** pode-se, a critério do usuário, incluir uma classe e estabelecer um relacionamento (agregação ou generalização) e assim sucessivamente.

⇒ ***4ª etapa:*** *definir a cardinalidade das correspondências entre as versões: como exemplo, as versões da subclasse Terrestre apresentam correspondência de 1:1 (por exemplo) em relação a superclasse Veículo.*

- *selecionar o ícone de cardinalidade*  ;
- *selecionar a subclasse com as versões descendentes (que, no exemplo, é a classe Terrestre) e confirmar com um clique;*
- *selecionar a superclasse com as versões ascendentes (que no exemplo é a classe Veículo); Caso a subclasse escolhida apresente uma única superclasse, a interface selecionará esta superclasse automaticamente, pulando este item;*
- *após a seleção da superclasse uma caixa de diálogo será apresentada com botões de seleção para ser escolhida uma das seguintes opções: 1:1, n:1, 1:n ou n:m;*

*A figura 4.14 mostra a definição da cardinalidade de correspondência entre a subclasse Terrestre e a sua superclasse Veículo.*

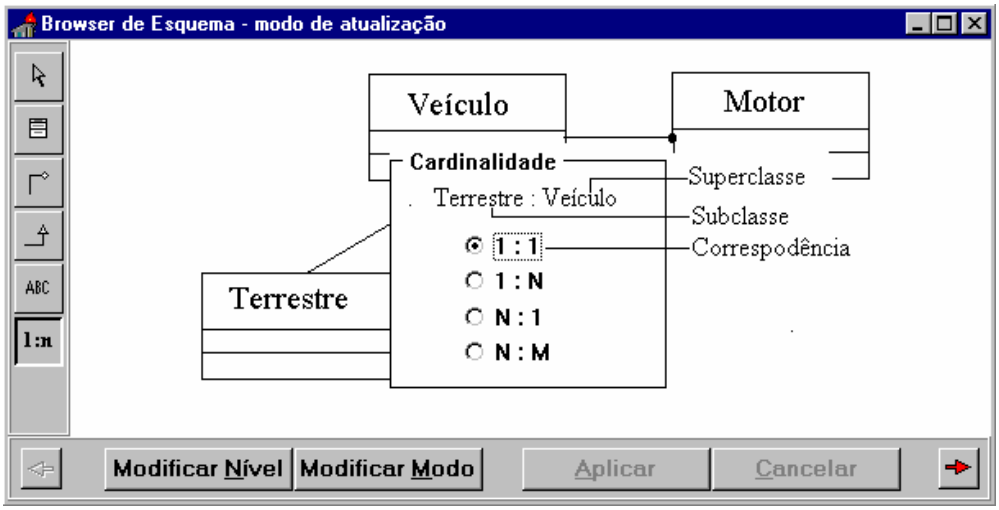



FIGURA 4.14- Definindo uma correspondência

*O layout do esquema da figura 4.14, após a definição da cardinalidade entre a subclasse Terrestre e a sua superclasse Veículo, fica semelhante ao layout mostrado na figura 4.15.*

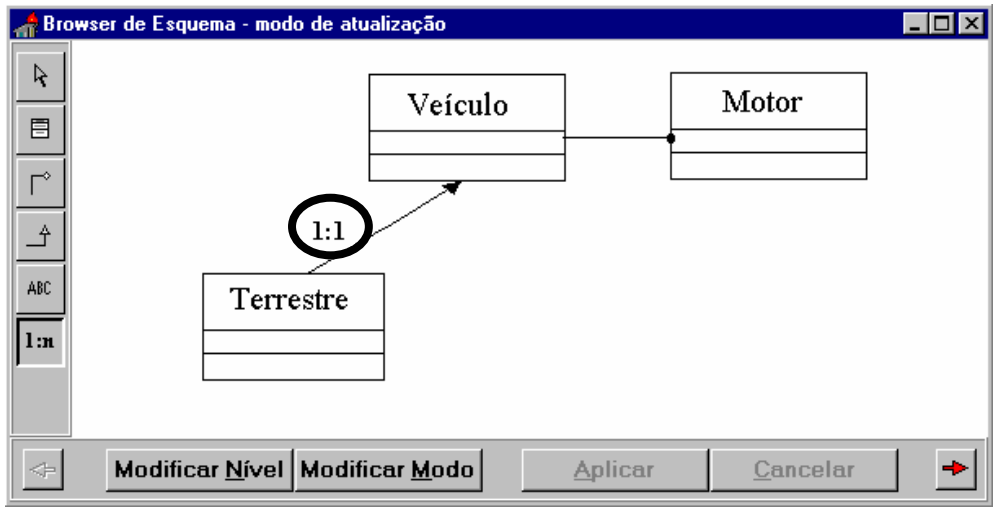



FIGURA 4.15- A correspondência entre a classe Terrestre e Veículo (1:1)

⇒ ***5ª etapa:*** definir os atributos de cada classe:

- cabe ressaltar, novamente, que os atributos podem ser manipulados através do Browser de Classe (descrito no item 4.5), que é o *browser* recomendado para essa funcionalidade.

- selecionar o ícone de atributos  e confirmar a opção, clicando em cima da classe desejada. A tela de especificação de atributos será ativada para que os atributos sejam cadastrados, como mostra a figura 4.16;


FIGURA 4.16- Definindo os Atributos da Classe *Terrestre*

### 4.4.4 A Alteração do Esquema

As ações de alteração do esquema de um banco de dados do SGBDOO $O_2$ (modo de atualização), que o usuário pode efetuar, são: (1) incluir uma nova classe no esquema; (2) alterar os atributos de uma classe; e (3) excluir uma classe existente.

As operações acima serão atualizadas, quando o usuário escolher a opção *Aplicar* (botão na barra de *Botões Específicos*) e confirmar a operação. O usuário deverá ter cuidado em alterar o esquema do BDO, quando a(s) classe(s) envolvida(s) já possuir(em) instâncias. A interface vai mostrar uma mensagem informativa antes de permitir as operações acima.

*Uma proposta futura é incorporar à interface o suporte, a evolução de esquemas, que está sendo desenvolvido atualmente no CPGCC [GAL 97]. Tal trabalho visa a definir um modelo de suporte à evolução de esquemas conceituais para bancos de dados orientados a objetos com o emprego de mecanismos de versões, particularmente, o modelo definido por Golendziner em [GOL95].*

*O modelo de evolução de esquemas é proposto como forma de guiar as alterações em um modelo de banco de dados orientado a objeto. O modelo apresentado é adaptável a vários sistemas de banco de dados orientados a objetos, pois as idéias propostas são apresentadas de forma genérica e não definidas para um sistema específico.*

*A proposta tem por objetivo conduzir as modificações realizadas em sistemas de banco de dados orientados a objetos, mantendo o histórico dessas alterações através da derivação de versões. O estado anterior às transformações é mantido, permitindo a*

*navegação retroativa e proativa entre as versões, para a realização de operações de alteração e consulta. O modelo assegura, ainda, a integridade das instâncias armazenadas no banco de dados em qualquer perspectiva de versão sob a qual são resgatadas.*

*O modelo prevê um mecanismo de modificação de esquema onde versões são utilizadas para armazenar o estado evolucionário do esquema, de suas classes e métodos e ainda para posterior adaptação das instâncias vigentes no banco de dados. A validade dos objetos armazenados no banco de dados e das alterações realizadas na estrutura e comportamento do esquema é garantida pela utilização de restrições de integridade, permitindo, ainda, a recuperação dos objetos sob qualquer perspectiva de versão.*

## 4.5 O Browser de Classe

O *layout* do browser de classe foi influenciado pelo layout da *Janela de Informações*, da interface OdeView [AGR 90], que apresenta a definição textual de uma classe em uma viewport e utiliza mais três viewports para mostrar as superclasses, subclasses e *links* para os objetos. O *layout* desse browser também teve influência no ObjectStore Inspector [VIV 96], que também utiliza viewports para apresentar as informações.

No browser de classe da interface optou-se por dividí-la em cinco *viewports* (subjanelas) distintas: (1) *viewport* Contexto; (2) *viewport* Superclasses; (3) *viewport* Subclasses; (4) *viewport* Atributos da Classe; (5) v*iewport* Objetos.

As *viewport*s funcionam como hiperdocumentos, com âncoras para as classes relacionadas. As âncoras são facilmente identificadas, pois aparecem sublinhadas. Por exemplo, na figura 4.17, ao clicar na superclasse *Veículo* (tanto na *viewport Contexto*, como na *viewport Superclasses*), o Browser de Classe atualizará todas as cinco *viewport*s para a nova classe selecionada, que, no exemplo, seria a Classe *Veículo*. A figura 4.17 mostra o *layout* do browser de classe com a definição de uma classe.

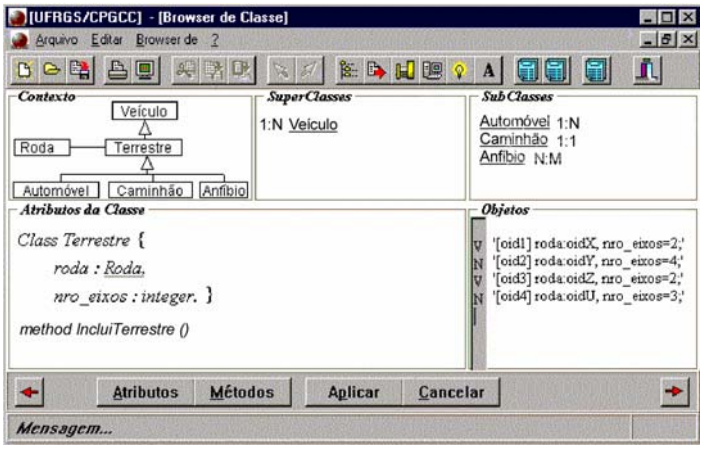



FIGURA 4.17- O layout do browser de classe

O Browser de Classe permite ao usuário visualizar ou atualizar (ver item 4.2) uma única classe por vez, ou seja, definir seus métodos e atributos (recomendado defini-los por esse browser).

### 4.5.1 A *viewport* Contexto

Essa *viewport* mostra, graficamente, o contexto da classe selecionada (destacando-a) com as respectivas superclasses, subclasses e classes agregadas. Os critérios para selecionar as classes que farão parte de um contexto são:

- as Superclasses do primeiro nível acima na hierarquia de classes, sendo para cada Superclasse:
    - as suas Superclasses até a classe raiz dessa ramificação na hierarquia de classes;
    - ***obs.: NÃO*** entram as suas subclasses e classes agregadas.

- as Subclasses do primeiro nível abaixo na hierarquia de classes, sendo para cada Subclasse:
    - as suas Subclasses até as folhas dessa ramificação na hierarquia de classes;
    - ***obs.: NÃO*** entram as suas superclasses e classes agregadas.

- as classes agregadas;

A figura 4.18 mostra o exemplo do contexto, da classe *Terrestre*, do esquema exemplo descrito na figura 4.17. Na viewport Contexto o usuário pode ir para as classes que são visualizadas nesta viewport (funciona com âncoras, *links*). Por exemplo, se o usuário clicar na classe Veículo (representada por um retângulo em volta), o browser de classe irá atualizar todas as demais viewports com as informações desta nova classe selecionada.

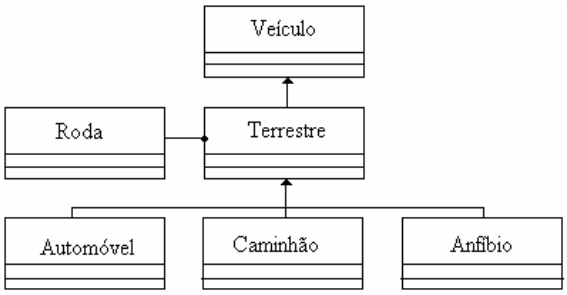



FIGURA 4.18- Um exemplo de contexto: classe *Aquático*

### 4.5.2 A *Viewport* Superclasses

Essa *viewport* apresenta uma lista com todas as Superclasses relacionadas com a classe selecionada, seguindo os critérios descritos acima. Cabe ressaltar que essa *viewport* funciona como um hiperdocumento, com *links* para as superclasses

relacionadas. A figura 4.19 mostra uma ampliação da viewport superclasses, da figura 4.17.

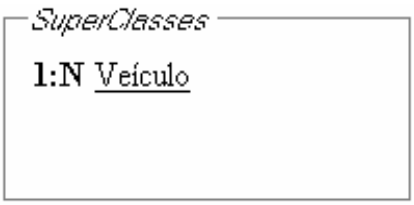



FIGURA 4.19- A *viewport* Superclasses

*Nessa viewport pode-se também visualizar as correspondências que existem entre a classe selecionada (ver figura 4.21, viewport atributos da classe) e suas superclasses. No exemplo a correspondência entre a classe Terrestre e a sua superclasse Veículo é de 1:N (ver seção 4.4.2, cardinalidade das versões).*

### 4.5.3 A *Viewport* Subclasses

Essa *viewport* apresenta uma lista com todas as Subclasses relacionadas com a classe selecionada, seguindo os critérios descritos acima. Cabe ressaltar que essa *viewport* funciona como um hiperdocumento, com *links* para as subclasses relacionadas.

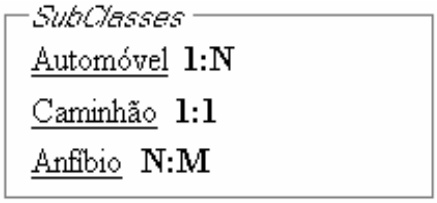



FIGURA 4.20- A *viewport* Subclasses

A figura 4.20 mostra um ampliação da viewport subclasses, da figura 4.17.

*Nessa viewport pode-se também visualizar as correspondências que existem entre as subclasses e a classe selecionada (ver figura 4.21, viewport atributos da classe). No exemplo a correspondência entre a subclasse Automóvel e a classe Terrestre é de 1:N (ver seção 4.4.2, cardinalidade das versões); a correspondência entre a subclasse Caminhão e a classe Terrestre é de 1:1; e a correspondência entre a subclasse Anfíbio e a classe Terrestre é de "n : m".*

### 4.5.4 A *Viewport* Atributos da Classe

Essa *viewport* apresenta uma descrição textual dos atributos e métodos da classe selecionada. As classes agregadas aparecem sublinhadas e funcionam como âncoras. Basta um simples clique para visualizar os atributos e métodos dessa classe agregada. A figura 4.21 mostra um ampliação da viewport classes, da figura 4.17.

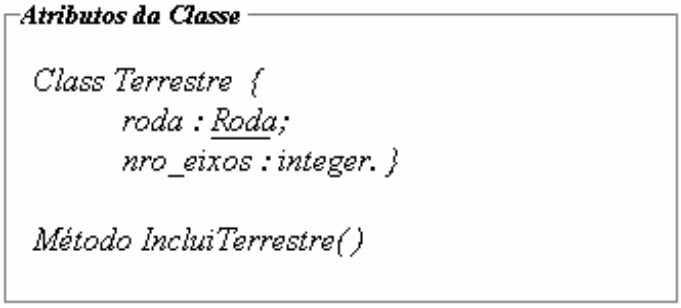



FIGURA 4.21- A *viewport* Atributos da Classes

Nessa viewport também é possível visualizar os métodos que a classe possui. A idéia em relação aos métodos é utilizar a sintaxe da linguagem utilizada pelo SGBDOO $O_2$, que é a linguagem $O_2C$. O browser de classes não irá permitir a inclusão de novos métodos, mas apenas a visualização. Quando o usuário desejar inserir novos métodos, a interface irá transferir essa tarefa para o SGBDOO $O_2$.

### 4.5.5 A *Viewport* Objetos

Essa *viewport* mostra a relação dos objetos (os primeiros atributos) da classe selecionada. O critério de escolha é os primeiros atributos do tipo texto, que couberem na *viewport objetos*, sem a necessidade da utilização de *scroll*. Esse critério foi escolhido pela simplicidade e facilidade de implementação.

Na viewport objetos também é mostrado um *status*, informando se o objeto é versionado: possui versão (**V**) ou não possui versão (**N**). Os objetos mostrados funcionam como âncoras (*links*), pois, com um simples clique, o Browser de Objeto será ativado com esse objeto por *default*, onde são exibidos os demais atributos do objeto, entre outras informações. A figura 4.22 mostra um ampliação da viewport objetos, da figura 4.17.

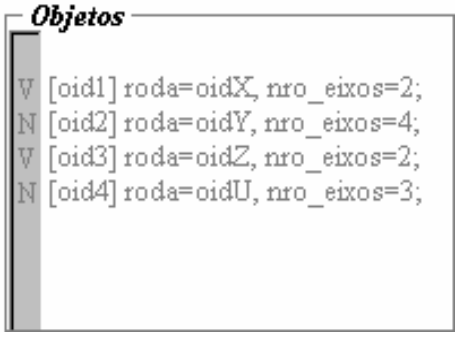



FIGURA 4.22- A viewport Objetos

## 4.6 O Browser de Objeto

O Browser de Objeto tem a finalidade de apresentar os objetos da classe, selecionada através do browser de classe e suas versões, permitindo sua manipulação (visualização ou atualização). A característica principal do browser - e da interface - é o suporte para versões.

Buscando manter uma padronização, o browser de objeto foi dividido em cinco *viewports*, sendo que as três primeiras *viewports* originaram-se do browser de classe. As outras duas *viewports*, próprias do browser de objeto são: (1) *viewport* Instância; e (2) *viewport* Grafo. A figura 4.23 mostra o layout do browser de objeto.

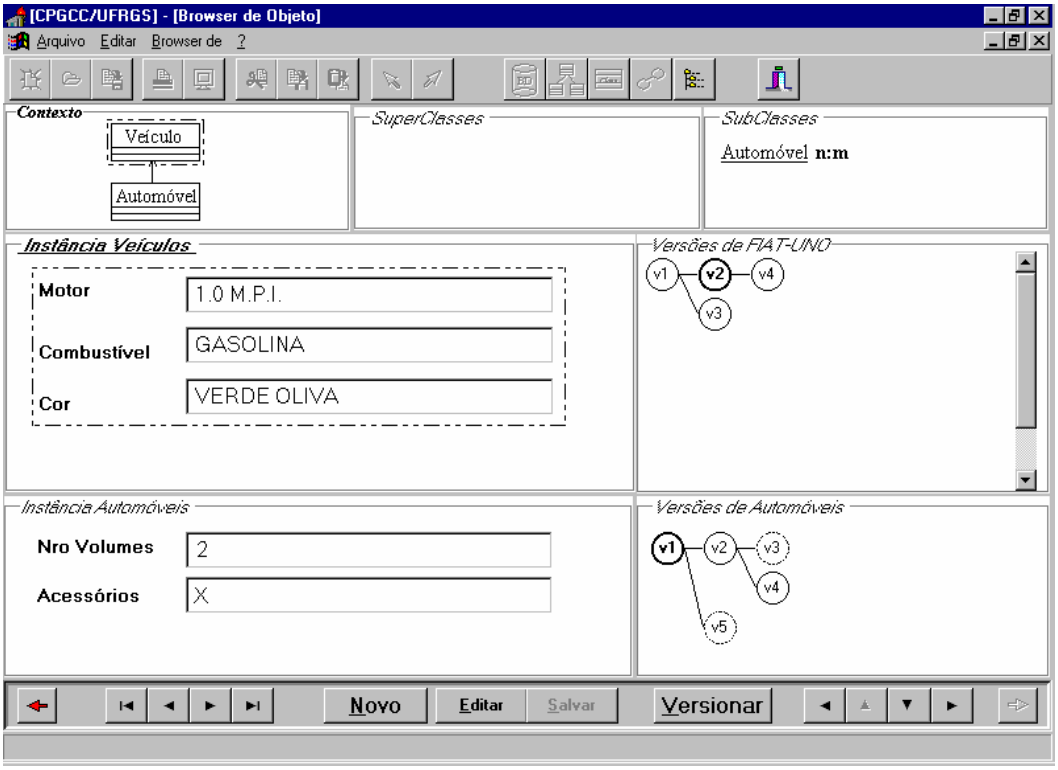



FIGURA 4.23- O *layout* do Browser de Objeto

A característica principal do browser de objeto, que merece um destaque maior, é a capacidade de visualizar e manipular as diferentes versões de um objeto e isto nos vários níveis da hierarquia de classes (herança por extensão). O browser trata uma versão como sendo um objeto, pois em [GOL 95] cada versão representa um estado identificável de um objeto, considerado pelo usuário como semanticamente significativo, devendo ser tratada como um objeto no modelo de dados.

Um problema a ser solucionado é que o SGBDOO $O_2$ [O2 96] possui herança por refinamento, onde os objetos e as versões são representadas no nível mais especializado da hierarquia de classes. Um estudo mais aprofundado faz-se necessário para descobrir uma solução adequada para o problema do versionamento no nível mais especializado *versus* o versionamento nos vários níveis da hierarquia de herança.

Golendziner [GOL 95] destaca que, sem a possibilidade de representar as versões nos vários níveis, facilidades do modelo de dados podem ser utilizadas, mas com certas desvantagens. Golendziner cita possíveis soluções para o versionamento no nível mais especializado e descreve uma possível implementação do versionamento de objetos nos vários níveis da hierarquia de herança.

### 4.6.1 A *Viewport* Instância

Esta *viewport* mostra os atributos do objeto, não-versionado ou versionado (versão atual). Optou-se por utilizar uma *viewport* semelhante à janela de visualização de atributos do SGBDOO $O_2$, já que essa apresenta características interessantes, tais como: caixas de entradas dos atributos com moldura; *links* para os atributos compostos ou a possibilidade de mostrar os "subobjetos" na mesma viewport; entre outras características.

Esta *viewport* poderá ser dividida em outras *viewport*s, dependendo do número de superclasses que a classe apresenta, ou seja, dentro da *viewport Instância* haverá uma *viewport* para mostrar os atributos da Classe selecionada e uma *viewport* para mostrar os atributos de cada uma das superclasses, se existirem. Os atributos das subclasses não terão uma *viewport*, ou seja, não serão visualizados na *viewport* Instância.

O critério adotado para nivelar as *viewports* do interior da *viewport Instâncias* é conforme o grau da classe na hierarquia de classes. Como exemplo, na figura 4.23 a classe selecionada *Anfíbio* (grau 3), possui duas superclasses (*Terrestre* e *Aquático*), ambas com o grau 2 e essas com uma superclasse *Veículo*, com o grau 1. Portanto, as *viewport*s com o mesmo grau deverão ficar com a mesma posição na tela, horizontalmente.

### 4.6.2 A *Viewport* Grafo

Esta *viewport* mostra um grafo que corresponde à representação gráfica das versões de um objeto versionado. O grafo é uma representação gráfica do objeto versionado [GOL 95]. A representação gráfica adotada para mostrar o Grafo com as diferentes versões do objeto versionado é a apresentada em [GOL 95].

Uma *viewport* grafo é apresentada para cada *viewport* instância. Um objeto versionado é composto por versões que são representadas graficamente por um círculo, formando um grafo de derivação.

### 4.6.3 A Barra de Status do Browser de Objeto

A barra de status, como a barra de ferramenta, pode conter botões de comando [MS 97]. Na interface a barra de status, do browser de objeto, apresenta dois conjuntos de botões para navegação: um à esquerda, e o outro à direita; além de outros botões de comando, conforme mostra a figura 4.24.


FIGURA 4.24- A barra de status do browser de objeto

O primeiro conjunto de botões, à esquerda, se faz necessário, pois permite ao usuário navegar pela classe como um todo, seguindo a ordem de criação dos objetos (não-versionados ou versionados(versão default) ). O segundo conjunto de botões, à direita, se faz necessário, pois permite ao usuário percorrer as diferentes versões (representadas graficamente pelo grafo de derivação) do objeto versionado atual. Outros quatro botões foram definidos para essa barra de status: *novo*, *editar/cancelar*, *salvar* e *versionar*. O botão *novo* se faz necessário, pois permite ao usuário criar um novo objeto (não-versionado). O botão *versionar* se faz necessário, pois permite ao usuário criar novas versões para o objeto versionado atual. Os botões *editar*/*cancelar* e *salvar* são necessários, pois permitem ao usuário atualizar as informações no banco de dados.

As opções disponíveis (botões) na barra de status, da janela do browser de objeto, com suas funcionalidades são:

- *primeiro conjunto de botões para navegação*, à esquerda da barra: esses botões permitem a navegação pelos objetos da classe. Os objetos podem ser *não versionados* (*nv*) ou *versionados* (*v*).

   : vai para o primeiro objeto da classe (*nv* ou *v*). É seguida a ordem de criação do objeto (*nv* ou *v*);

   : vai para o objeto (*nv* ou *v*) anterior ao objeto (*nv* ou *v*) corrente;

   : vai para o objeto (*nv* ou *v*) posterior ao objeto (*nv* ou *v*) corrente;

   : vai para o último objeto da classe (*nv* ou *v*).

- *segundo conjunto de botões para navegação*, à direita da barra: esses botões permitem a navegação pelas versões do objeto versionado (grafo em destaque).

   : vai para a versão acima da versão corrente (mesma versão de origem);

   : vai para a versão anterior a versão corrente;

   : vai para a versão posterior a versão corrente;

   : vai para a versão abaixo da versão corrente (mesma versão de origem);

  ⇒ o critério de navegação pelas versões obedece à ordem de derivação das versões juntamente com o histórico de criação das versões, de cada objeto versionado. À medida que as versões vão sendo criadas, elas recebem um número seqüencial crescente, que inicia com o número um (1..n).

O critério de navegação pelas versões e as funcionalidades dos botões, para essa navegação, ficam mais claros com o exemplo da figura 4.25.

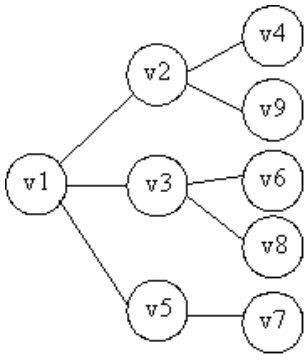



FIGURA 4.25- Um Grafo Exemplo

| *Estando na versão v1, se clicar em:* | *Estando na versão v2, se clicar em:* |
|---|---|
| ▲ : *desabilitado* (fica em *v1*); | ▲ : *desabilitado* (fica em *v2*); |
| ◄ : *desabilitado* (fica em *v1*); | ◄ : volta para *v1*; |
| ► : vai para *v2*; | ► : vai para *v4*; |
| ▼ : *desabilitado* (fica em *v1*); | ▼ : vai para *v3*; |

| *Estando na versão v3, se clicar em:* | *Estando na versão v8, se clicar em:* |
|---|---|
| ▲ : vai para *v2*; | ▲ : vai para *v6*; |
| ◄ : volta para *v1*; | ◄ : volta para *v3*; |
| ► : vai para *v6*; | ► : *desabilitado* (fica em *v8*); |
| ▼ : vai para *v5*; | ▼ : *desabilitado* (fica em *v8*); |

- botão Novo : cria um novo objeto (não versionado);
  ⇒ o usuário informa os atributos e depois pressiona o botão salvar;

- botão Versionar: transforma um objeto não versionado em versionado, ou cria mais uma versão derivada.
  ⇒ utilizar o objeto (não versionado ou versionado) corrente ou escolher outro, através do *primeiro conjunto de botões para navegação*. Essa escolha deve ser feita antes de pressionar o botão *Versionar*.
  - ♦ Se o objeto selecionado é não versionado (primeira versão):
    a) pede-se um nome para o objeto versionado: (o nome para o objeto versionado é opcional). Nesse momento é criado um objeto versionado;
    b) o objeto corrente (escolhido) passa a ser a *primeira versão* do objeto versionado;
    c) pede-se um nome para a versão. O nome para a versão é opcional. O nome default é a letra "v" mais o número da versão (*v1*, ..., V*n*).

d) *estabelecer uma derivação:* cria-se (automaticamente) uma cópia da *versão1*. Essa cópia passa a ser a *versão2* (idem ao item "c"). A *versão2* para a ser uma versão derivada da *versão1*.

- Se o objeto selecionado é versionado (mais uma versão derivada):
  e) *estabelecer uma derivação:* escolher uma versão para ser a versão de origem, ou seja, a versão origem da derivação.
     - permitir a escolha dessa versão no grafo de derivação; *ou;* utilizar a versão default (essa escolha é feita antes de pressionar o *botão Versionar*);
     - cria-se (automaticamente) um cópia da versão escolhida. (idem ao item "c" do tópico anterior). Essa cópia passa a ser uma versão derivada da versão escolhida;

### 4.6.4 Construindo um objeto versionado

Para ficar mais claro o entendimento das características do Browser de Objeto e o seu suporte para versões, será exemplificada, passo a passo, a construção do objeto versionado ***fiat-uno***, representado pela figura 4.26:

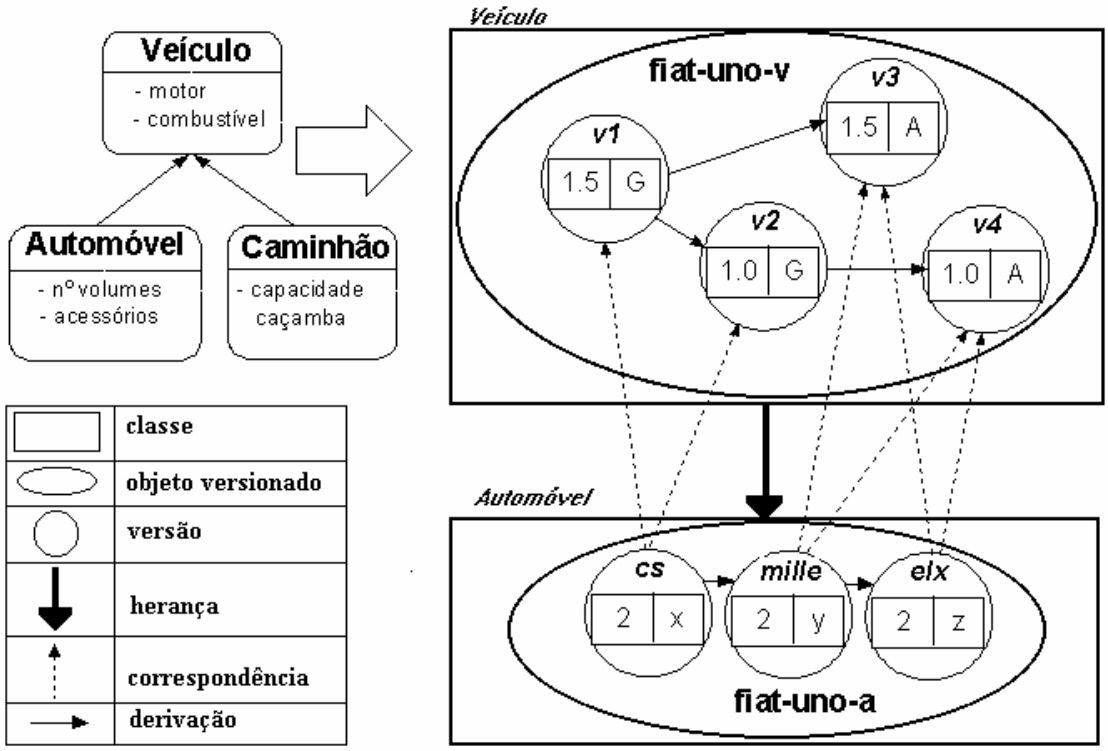



FIGURA 4.26- Representação do objeto versionado *fiat-uno* em mais de um nível da hierarquia de classes e suas correspondências [GOL 95]

O objeto versionado ***fiat-uno*** será construído pelo processo *top-down*. Primeiro será definido no nível de *Veículo* e depois no nível de *Automóvel*. O objeto que aparece na figura 4.27 é a primeira instância da classe *Veículo* (objeto não versionado). Para ele ser incluído, foi pressionado o botão *Novo* e após foi informado o conteúdo para cada um dos seus atributos.

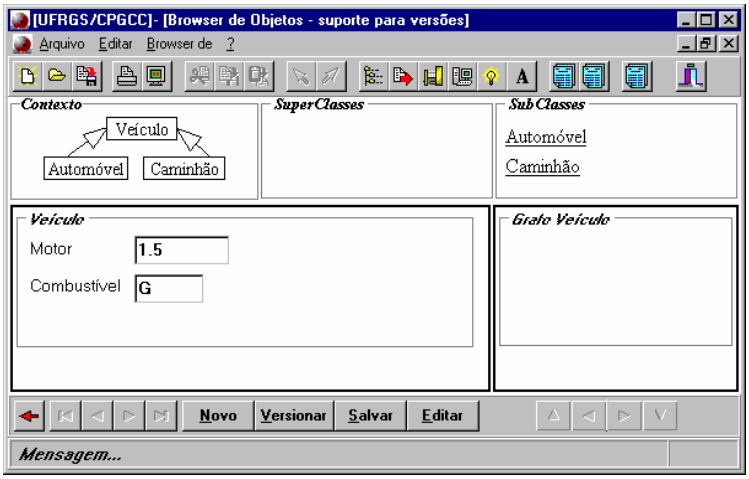



FIGURA 4.27- A primeira instância da classe *Veículo* (objeto não-versionado)

Como o objeto acima é não versionado, a próxima ação é versioná-lo. Esse processo é iniciado quando pressionamos o botão *Versionar*. Nesse momento é aberta uma caixa de diálogo (figura 4.28), pedindo que seja informado um nome para representar o *Objeto Versionado*, que, no exemplo, é ***fiat-uno*.**

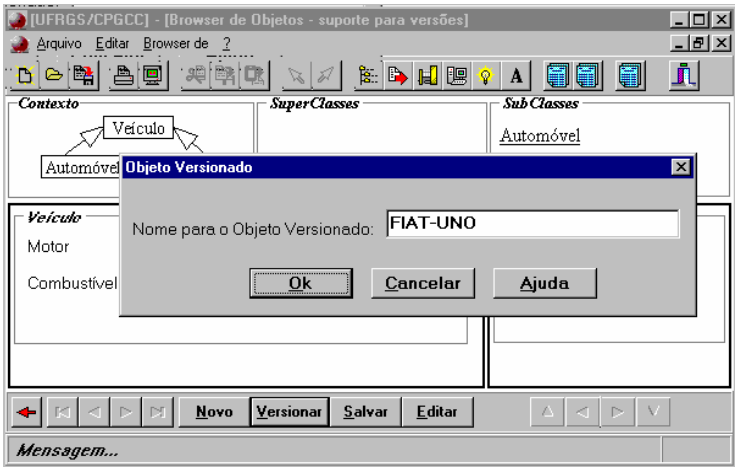



FIGURA 4.28- Caixa de diálogo para informar um nome para o objeto versionado

Seguindo os passos descritos na seção 4.6.3 (botão *Versionar*), o próximo é o *item b*. Depois de informado o nome para a *versão1* (item c), ou mantido o nome default, a interface estabelecerá automaticamente uma derivação entre a versão 1 e a versão 2 (*v1* → *v2)*, descrito no item *d*. Depois de estabelecida a derivação e informado o nome para

a versão 2 (ou mantido o nome default), editam-se os atributos, que, inicialmente, são uma cópia dos atributos da versão 1.

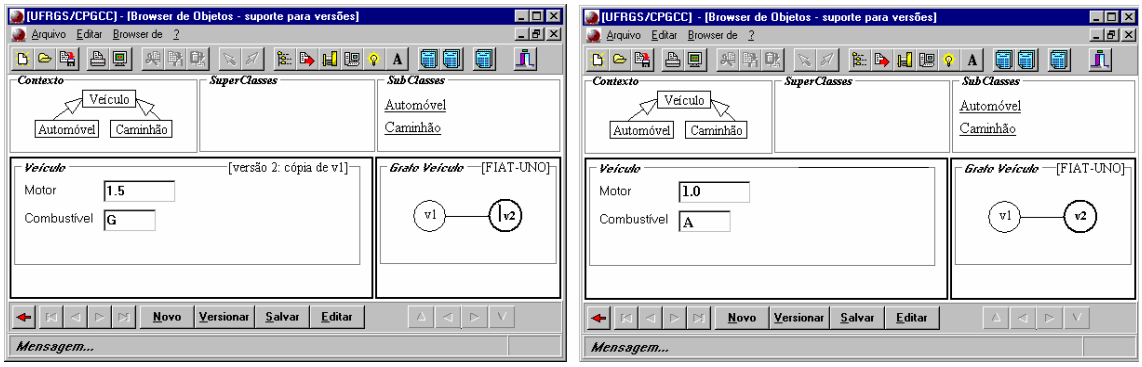



FIGURA 4.29- Estabelecendo uma derivação entre *v1* → *v2*

Um esboço da classe *Veículo*, após a inclusão da versão 2, ficaria assim:

TABELA 4.1- Um esboço da classe *Veiculo* com a inclusão da versão 2

| | **OID (identificador de objeto)** | | | **Atributos da classe *Veículo*** | | |
|---|---|---|---|---|---|---|
| | *id-entidade* | *classe* | *número da versão* | *Nome da Versão* | | |
| **1** | Fiat-uno | Veiculo | 1 | ***v1*** | **1.5** | **G** |
| **2** | Fiat-uno | Veiculo | 2 | ***V2*** | **1.0** | **G** |

A próxima etapa é incluir a terceira versão para o objeto versionado ***fiat-uno***. O processo inicia, pressionando-se o botão *Versionar*, e seguindo as instruções descritas no *item e*. Como será criada uma nova versão para um objeto já versionado, o primeiro passo é estabelecer uma derivação, escolhendo uma versão para ser a versão de origem. A interface estabelece automaticamente esse processo de derivação, cabendo ao usuário apenas escolher no grafo a versão de origem (essa escolha é feita antes de pressionar o botão *Versionar*). A partir desse ponto, a interface gera automaticamente uma cópia da versão de origem, que passa a ser a versão derivada que, no exemplo, é representada pela versão 3

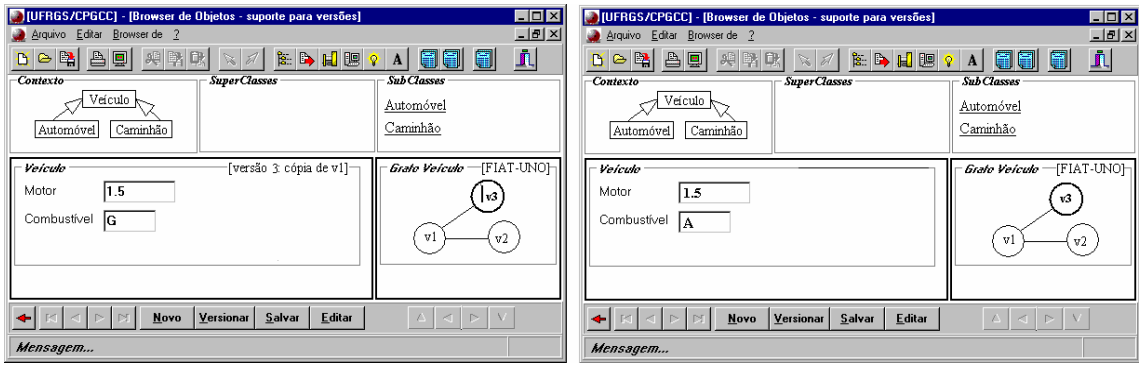



FIGURA 4.30- Estabelecendo uma derivação entre *v1* → *v3*

Um esboço da classe *Veículo*, após a inclusão da versão 3, ficaria assim:

TABELA 4.2- Um esboço da classe *Veículo* com a inclusão da versão 3

| | **OID (identificador de objeto)** | | | **Atributos da classe *Veículo*** | | |
|---|---|---|---|---|---|---|
| | *id-entidade* | *classe* | *número da versão* | *Nome da Versão* | | |
| **1** | fiat-uno | Veiculo | 1 | ***v1*** | **1.5** | **G** |
| **2** | fiat-uno | Veiculo | 2 | ***v2*** | **1.0** | **G** |
| **3** | fiat-uno | Veiculo | 3 | ***v3*** | **1.5** | **A** |

A inclusão da versão 4 para o objeto versionado *fiat-uno*, segue os mesmos passos descritos para incluir a versão 3.

O objeto ***fiat-uno*** foi incluído na hierarquia de classes no nível de V*eículo*, e agora será criado no nível de *Automóvel*. Com o layout da figura 4.30, será ativada a subclasse *Automóvel* no Browser de Objeto, escolhendo-a na *viewport Subclasses* ou na *viewport Contexto*, com um clique. O browser de objeto será restaurado com mais duas *viewports*: "*Automóvel*" e "*Grafo Automóvel*", com o objeto corrente da superclasse *Veículo*, que, no exemplo, é o objeto versionado *fiat-uno*. Como o objeto versionado *fiat-uno* não apresenta nenhuma versão no nível de *Automóvel*, as *viewport*s apareceram vazias. A figura 4.31 ilustra isto.


FIGURA 4.31- Objeto versionado *fiat-uno* no nível de Automóvel

A versão a ser criada é uma versão na subclasse, e as instruções descritas na seção 4.6.3, para o botão *Versionar*, sofrem algumas alterações, como segue:

- Se o objeto selecionado é não versionado (primeira versão):

  a) o objeto versionado é o objeto corrente da superclasse, que, no exemplo, é o objeto versionado *fiat-uno*;
  b) cria-se, automaticamente, uma versão, com os atributos vazios (ver figura 4.31);
  c) pede-se um nome para a versão. O nome para a versão é opcional. O nome default é a letra "v" mais o número da versão (*v1*, ..., V*n*).
  d) *estabelecer uma correspondência* (na seção 4.6.3, nesse item d, estabelece uma derivação): essa correspondência obedece à

cardinalidade informada no momento da construção do esquema, descrito na seção 4.4.2, botão *Cardinalidade*. O usuário deve escolher no grafo da *superclasse* (que, no exemplo, é o grafo *Veículos*) a versão (ou versões) correspondentes.

No item *b* foi criada automaticamente uma versão, que será a primeira versão para o objeto versionado ***fiat-uno***, no nível de *Automóvel*. No item *c* deve ser informado um nome para a versão, que, no exemplo, será a versão "*cs*". A figura 4.32 ilustra isso.

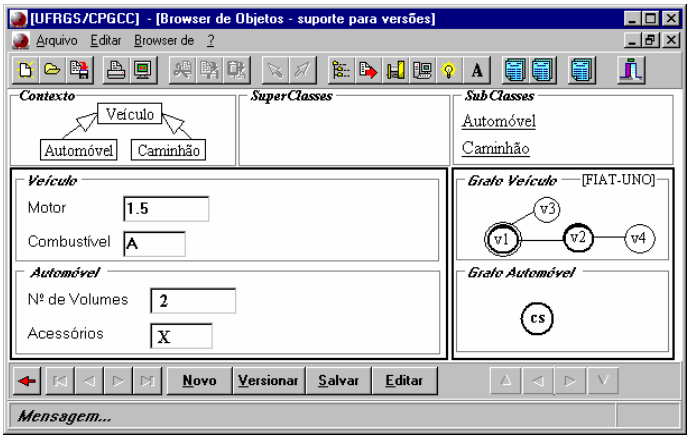



FIGURA 4.32- Objeto versionado *fiat-uno* no nível de Automóvel

A inclusão das versões "*mille*" e "*elx*" para o objeto versionado *fiat-uno*, no nível de *Automóvel* segue os mesmos passos descritos para incluir a versão "*cs*".

Na *viewport Grafo Veículo*, da figura 4.32, as versões *v1* e *v2*, aparecem destacadas, pois apresentam uma correspondência com a versão *cs* no nível de *Automóvel*. A versão *v1* aparece duplamente destacada, porque os seus atributos estão sendo visualizados na *viewport Veículo*.

## 4.7 O Browser de Consulta

O browser de consulta tem a finalidade de apresentar um subconjunto dos objetos (não versionados ou versionados) de uma classe ou um subconjunto das versões de um *objeto versionado*.

Idealmente, buscando manter uma padronização, o Browser de Consulta deverá apresentar características do Browser de Objeto, permitindo ao usuário especificar uma consulta e manipular (visualizar e atualizar) o subconjunto de objetos ou versões selecionadas. É possível criar novos objetos ou novas versões (de um objeto versionado), através desse browser, e essas instâncias deverão ser tornadas persistentes.

### 4.7.1 O Browser de Consulta: Modo Visual

O Browser de Consulta, modo visual, permite aos usuários restringirem diretamente a exibição dos objetos ou versões de um objeto versionado, com a

existência de filtros (predicados) na estrutura do browser de objeto em qualquer nível da hierarquia de classes.

Nesse modo visual, o usuário informa predicados (valores) na caixa de edição do(s) atributo(s) desejado(s) e a interface converte esses filtros em linguagem SQL, e o SGBDOO processa a consulta e retorna um subconjunto de objetos que satisfazem o critério de seleção.

Com os objetos retornados, a interface deverá mostrar visualmente ao usuário esse processo de retorno, ou seja, destacar no grafo de versões apenas as versões que obedecem aos critérios de seleção.

O browser de consulta pode ser ativado pelo browser de objeto (recomendado) ou por um dos outros browsers da interface. Estando no browser de objeto, o usuário ativa o Browser de Consulta e a interface apresenta o Browser de Objeto com as mesmas classes, mas com as caixas de edição de cada atributo em branco, para que o usuário informe os predicados (valores) de filtro que satisfaçam sua consulta, como mostra a figura 4.33.

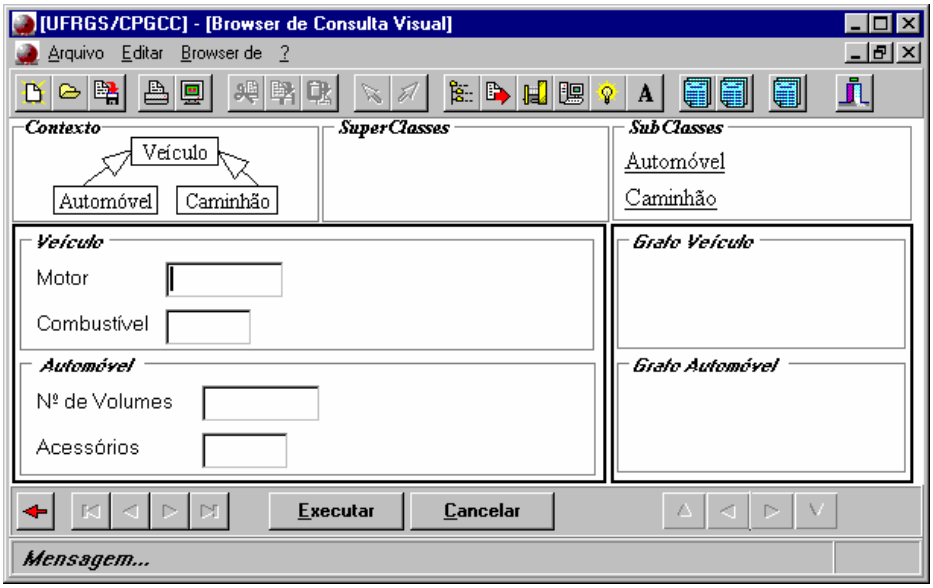



FIGURA 4.33- O Browser de Consulta

A diferença básica de *layout* entre o Browser de Consulta e o Browser de Objeto ocorre da barra de status (ver figura 4.24). No modo de consulta, os botões (Novo | Versionar | Salvar | Editar) são substituídos pelos botões (Executar | Cancelar).

Um aspecto importante a ser ressaltado é sobre quais objetos (não-versionados ou versionados (versões) ) a consulta será processada. Por *default*, as consultas serão processadas sobre as versões do objeto versionado ativo. Caso o usuário deseje consultar os objetos (não-versionados) da classe como um todo, deverá especificar isto, clicando com o botão direito do mouse sobre o botão *executar*, onde será visualizado um menu suspenso, e escolher essa opção. As opções do menu suspenso são: '*Consultar Versões*' e '*Consultar Objetos Não-versionados*'.

- *se o objeto ativo é Objeto Não-Versionado:* os predicados (filtros), informados nas caixas de edição dos campos, afetarão os objetos da classe como um todo, ou seja, o usuário vai navegar pelos objetos não versionados (nv) e versionados, e não pelas versões de um objeto versionado específico.

- *se o objeto ativo é Objeto Versionado:* a consulta afetará somente as versões do objeto versionado ativo. A *viewport Grafo* continuará mostrando todas as versões do objeto versionado, mas somente estarão disponíveis as versões que satisfizerem os predicados informados. A interface mostrará, em formato normal, as versões que pertencem ao subconjunto pretendido, e as versões que não pertencerem serão mostradas no formato inativo (em tom de cinza).

Para ficar mais claro o entendimento do Browser de Consulta, modo Visual, serão demonstrados os passos necessários para a realização da seguinte consulta: "Obter as versões do *objeto versionado* ***fiat-uno*** que utilizam Gasolina como combustível". Estando com o *layout* da figura 4.33, primeiramente ativa-se o Browser de Consulta. O próximo passo é informar os predicados (valores) para o(s) atributo(s) que se deseja filtrar. No exemplo, o atributo é "*Combustível*" e o predicado é "*G*", ou seja, digita-se, na caixa de edição do atributo "*Combustível*", o valor "*G*" e pressiona-se o botão "*Executar*". O layout da consulta será semelhante ao mostrado pela figura 4.34.

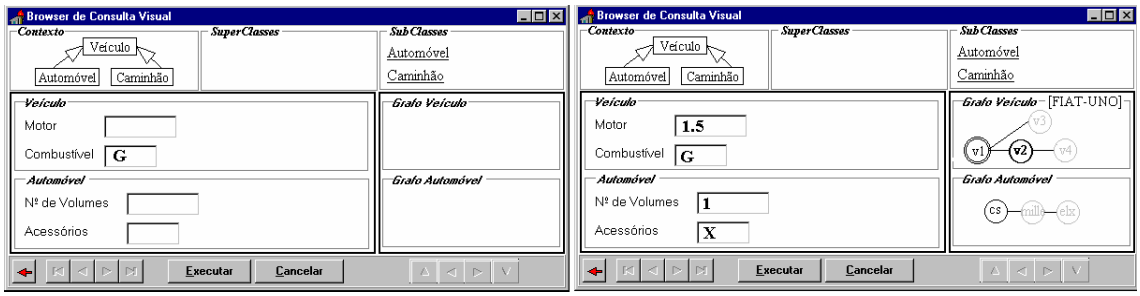



FIGURA 4.34- Selecionando versões do objeto versionado *fiat-uno* - consulta 1

A consulta SQL similar para realizar a consulta visualizada na figura 4.34 seria:

*Select **
*From Veículo_Versões*
*Where Veículo_Versões.Combustível = 'G'*

Será criada uma consulta, envolvendo o objeto versionado *fiat-uno* nos dois níveis da hierarquia de classes: no nível de *Veículo* e no nível de *Automóvel.* "Obter as versões do *objeto versionado* ***fiat-uno*** que utilizem Álcool como combustível (nível de Veículo) e apresentem 2 volumes (nível de Automóvel)". A figura 4.35 mostra o resultado dessa consulta.

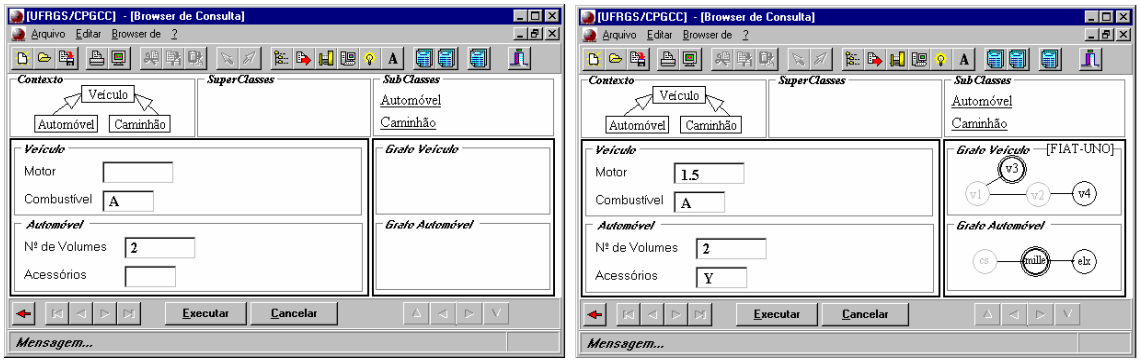



FIGURA 4.35- Selecionando versões do objeto versionado *fiat-uno* - consulta 2

A consulta SQL similar para realizar a consulta visualizada na figura 4.35 seria:

*Select* *
*From* *Veículo_Versões, Automóvel_Versões*
*Where* *Veículo_Versões.OID = Automóvel_Versões.OID*
*and Veículo_Versões.Combustível = 'A'*
*and Automóvel_Versões.Volumes = 2*

### 4.7.2 Utilização de Predicados

A utilização de predicados torna-se necessária em consultas que envolvam comparações mais elaboradas, do que simplesmente uma comparação de igualdade, onde: *atributo* "é igual a" *valor* (Veículo.Combustível = 'G').

Basicamente, uma consulta envolvendo a utilização de predicados é composta de três partes: (1) Atributo; (2) Operador; e (3) Filtro(valor).

(1) Atributo: é um dado membro da classe (o próprio atributo do objeto). No exemplo da figura 4.33 os atributos da classe *Veículo* são: *Motor* e *Combustível*; e os atributos *Automóvel* são: *N.º de Volumes* e *Acessórios*.

(2) Operador: é um símbolo representado por um operador de comparação e/ou representado por um operador lógico.

- Operadores de comparação:
  - = : igual
  - < : menor,
  - > : maior;
  - ≤ : menor e igual;
  - ≥ : maior e igual;
  - ≠ : diferente;
- Operadores lógicos:
  - AND;
  - OR;
  - NOT;

(3) Filtro: é um valor que será utilizado para comparação (através dos operadores), com os valores armazenados nos atributos de objetos. Esse valor serve de filtro para selecionar versões de um objeto ou selecionar objetos de uma classe.

A figura 4.36 mostra as três partes básicas de uma consulta, envolvendo a utilização de predicados.

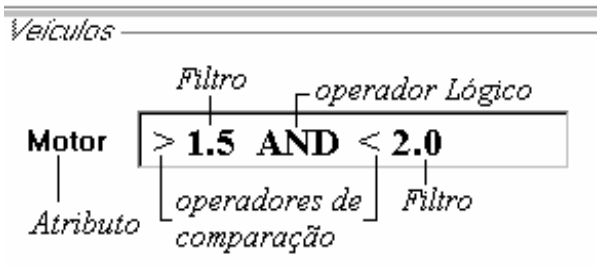



FIGURA 4.36- Estrutura básica de um consulta com predicados

Na figura 4.36, o atributo envolvido na consulta é o atributo *Motor*, pertencente à classe *Veículo*. Os operadores utilizados no exemplo são: operadores de comparação: >, <; e o operador lógico *AND*. Os valores de filtro sobre o atributo *Motor* é '1.5' e '2.0', isto é, filtrar da classe *Veículo* todas versões (objeto versionado ativo) onde o atributo *Motor* seja maior que '1.5' e menor que '2.0'. Na figura 4.36, a consulta similar em SQL seria:

```
Select  *
From  Veículo_Versões, Automóvel_Versões
Where Veículo_Versões.OID = Automóvel_Versões.OID
      and Veículo_Versões.Motor > '1.5'
      AND Veiculo_Versões.Motor < '2.0'
```

Observou-se que é mais intuitivo apresentar as informações do banco de dados ao usuário, agrupando-as por características semelhantes. Para isto, surge a necessidade de se criarem critérios de afinidades. Sugere-se apresentar as informações de um banco de dados através de browsers específicos, pois cada um trata de aspectos particulares da informação global. Os browsers sugeridos são: (1) browser de banco de dados: para as características relacionadas com a abertura e criação do banco de dados; (2) browser de esquema: para as características envolvidas com o projeto do esquema do banco de dados; (3) browser de classe: para as características que dizem respeito a uma classe individualmente, com seus métodos e atributos; (4) browser de objeto: para as características relacionadas com a manipulação dos objetos e suas versões; e (5) browser de consulta: para as características que envolvem consultas às informações do banco de dados.

# 5 Um protótipo da interface

Este capítulo apresenta um protótipo da interface com algumas das características descritas no seu projeto. A funcionalidade, mais explorada no protótipo, foi o *Browser de Objeto* e o seu suporte para o modelo de versões [GOL 95]. Também foram implementadas as características principais para o funcionamento do *Browser de Consulta*.

O projeto da interface, inicialmente, foi voltado para o suporte direto a bancos de dados do SGBDOO $O_2$. A idéia inicial é poder criar, ou ler, um banco de dados do $O_2$, através da interface. Isso permitirá ao usuário projetar o esquema do banco de dados, graficamente, além de permitir a manipulação dos objetos e conseqüentemente das diferentes versões do objeto. Também possibilitar ao usuário consultar graficamente, através de um *browser de consulta*, os objetos e suas versões. O projeto da interface também prevê o suporte para outros SGBDOOs, apenas para visualização (esquema e instâncias), que suportem o padrão ODMG-93 [CAT 96].

Inicialmente, o protótipo seria desenvolvido, utilizando o ambiente de programação visual da Microsoft®, o *Visual* $C^{++}$, para plataforma PC, sobre o Sistema Operacional Windows® NT. Seria utilizado esse ambiente, pois com ele se consegue a comunicação com a versão do SGBDOO $O_2$, para Windows® NT. Não foi possível conseguir uma versão do SGBDOO $O_2$, para o Sistema Operacional Windows® NT, com tempo hábil para se desenvolver o protótipo projetado inicialmente. Como alternativa foi desenvolvido um protótipo, utilizando o ambiente de programação da Borland®, o Delphi, para plataforma PC. Esse ambiente foi escolhido pelos seus vastos recursos à prototipação de uma interface.

## 5.1 Os recursos implementados para o Browser de Objeto no protótipo

Um dos principais objetivos do projeto da interface é o suporte para o modelo de versões [GOL 95]. Por esse motivo, centralizaram-se os esforços no desenvolvimento de um protótipo com essa característica: manipular (visualizar e atualizar) os objetos e suas versões. Das características apresentadas pelo modelo de versões e dos recursos projetados para o *Browser de Objeto* foram implementadas no protótipo as seguintes características, entre outras:

(1) suporte para a inclusão de objetos;

(2) suporte para o versionamento de objetos e inclusão de novas versões;

(3) suporte para o versionamento nos vários níveis da hierarquia de classes;

(4) suporte para a representação gráfica do objeto versionado, através do grafo de versões, permitindo a navegação pelo grafo e escolha de versões graficamente;

(5) suporte para a navegação pelos objetos (não-versionados e versionados) da classe como um todo;

(6) suporte para a navegação pelas diferentes versões de um objeto versionado.

## 5.2 O *Layout* geral do protótipo

No protótipo não foram implementadas todas as características propostas no projeto da Interface (ver capítulo 4), mas as funcionalidades mínimas de cada Browser, para acionar o Browser de Objeto e o Browser de Consulta.

O protótipo, por *default*, disponibiliza apenas o ícone de acesso para o Browser de Banco de Dados, pois uma sessão de trabalho somente pode ser iniciada depois da abertura de um banco de dados. A figura 5.1 mostra o *layout* geral do protótipo.

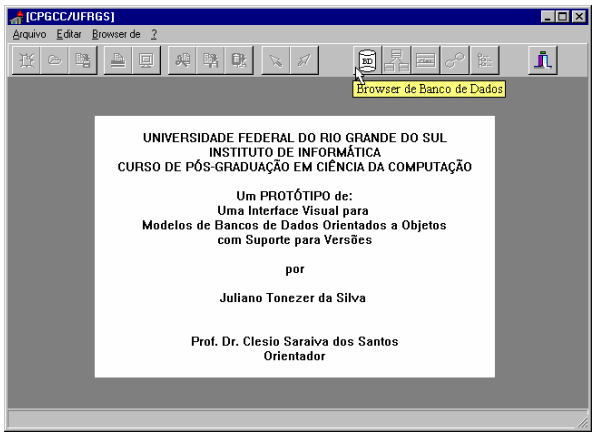



FIGURA 5.1- O layout do protótipo: janela principal

A figura 4.2, do capítulo anterior apresenta a hierarquia dos browsers e os caminhos (fluxo de acesso) que podem ser percorridos para ativar um browser. Os demais browsers só podem ser ativados depois que um banco de dados for aberto através do browser de banco de dados.

O browser de banco de dados pode ser ativado de duas maneiras: (1) pressionando o botão rápido, na barra de ferramentas, como aparece em destaque na figura 5.1, ou; (2) através da barra de menu: Browser de | Banco de Dados. A figura 5.2 mostra o browser de banco de dados, com um ícone de atalho que representa um banco de dados exemplo.


FIGURA 5.2- Protótipo: o *layout* do Browser de Banco de Dados

O browser de esquema pode ser ativado de três maneiras: (1) pressionando o botão rápido, na barra de ferramentas, como aparece em destaque na figura 5.2; (2) através da barra de menu: Browser de | Esquema, ou; (3) pressionando o botão Abrir, que aparece na barra de *status* do browser de banco de dados.

A figura 5.3 mostra o browser de esquema, com as classes do banco de dados, destacando por *default* a classe com maior hierarquia. A classe *Automóvel* pode ser destacada com um simples clique em qualquer área do retângulo que a representa. A classe que estiver em destaque será a *default* do browser de classe.

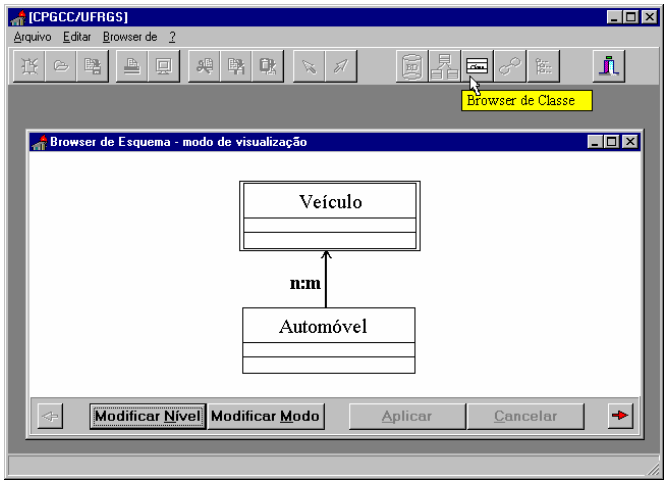



FIGURA 5.3- Protótipo: o *layout* do Browser de Esquema

O browser de classe pode ser ativado de três maneiras: (1) pressionando o botão rápido, na barra de ferramentas, como aparece em destaque na figura 5.3; (2) através da barra de menu: Browser de | Classe, ou; (3) pressionando o botão  , que aparece na barra de *status* do browser de esquema. A figura 5.4 mostra o browser de classe, com uma descrição textual da classe *Veículo* (que estava destacada no browser de esquema). A classe que estiver destacada neste browser de classe será a *default* no browser de objeto.

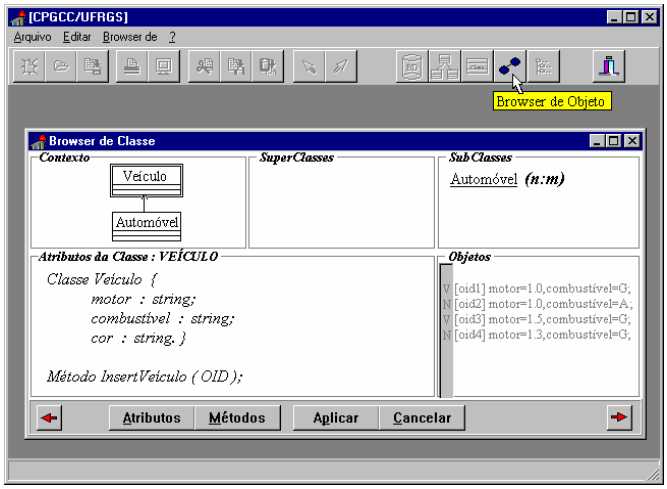



FIGURA 5.4- Protótipo: o *layout* do Browser de Classe

O **Browser de objeto** pode ser ativado de três maneiras: (1) pressionando o botão rápido, na barra de ferramentas, como aparece em destaque na figura 5.4; (2) através da barra de menu: Browser de | Objeto, ou; (3) pressionando o botão  , que aparece na barra de *status* do browser de classe. A figura 5.5 mostra o browser de objeto, com a classe *Veículo* (que estava destacada no browser de classe). Como a classe *Veículo* ainda não possui nenhum objeto (não-versionado ou versionado), os *grupos de botões para navegação* e o botão *Versionar* aparecem desabilitados. O botão *Salvar* somente será habilitado, quando se desejar incluir um novo objeto (não-versionado ou versionado) ou editar os atributos de um objeto já existente.

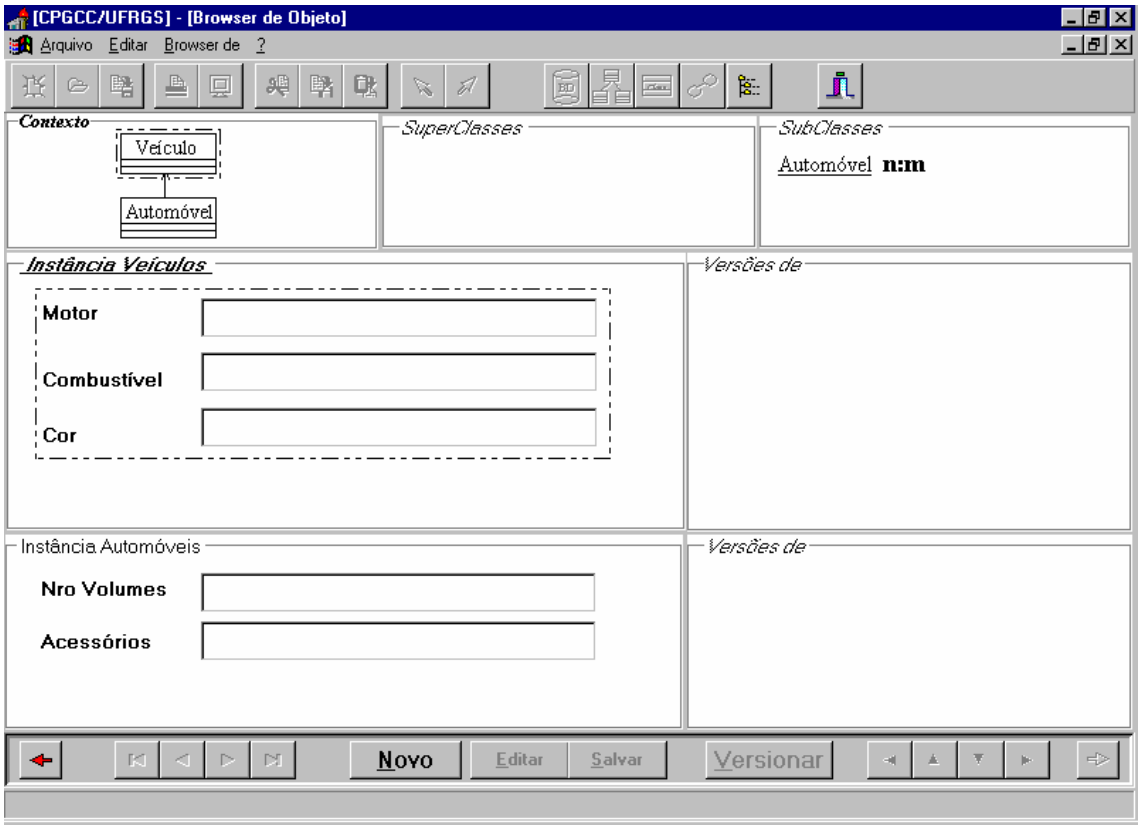



FIGURA 5.5- Protótipo: layout do Browser de objeto

Algumas considerações sobre o funcionamento do Browser de Objeto, que podem ser visualizadas na figura 5.5., devem ser mencionadas para ficar mais claro o entendimento deste browser:

(1) As operações disponíveis para o usuário, apresentadas na *barra de status* deste browser (ver item 4.6.3, figura 4.24), serão efetuadas sobre a *Classe Ativa*, ou seja, a classe que está em destaque. Por default, a classe ativa será a que estava em destaque no Browser de Classe. No exemplo da figura 5.5, a classe ativa é a *Classe Veículo*. O usuário identifica a classe ativa através de três critérios escolhidos para destaque:

(a) na *viewport Contexto* a classe ativa será destacada através de um retângulo pontilhado.

(b) na *viewport Instância* os atributos da classe ativa serão envoltos por um retângulo pontilhado.

(c) o título da *viewport Instância*, da classe ativa, estará com a sua fonte no seguinte formato: negrito, itálico e sublinhado.

(2) O usuário pode destacar uma classe, ou seja, torná-la Ativa, com um simples clique sobre a classe desejada: (a) na *viewport Contexto*; ou (b) na *viewport Instância*; ou (c) na *viewport SuperClasses*; ou (d) na *viewport Subclasses*;

(3) no exemplo da figura 5.5, a classe ativa é a classe *Veículo*. Portanto, ao clicar no botão *Novo*, esta operação afetará a classe *Veículo*.

### 5.3 Um estudo de caso

Para apresentar as características (ver item 5.1) implementadas no protótipo para o Browser de Objeto e deixar mais claro o entendimento destes recursos, optou-se por descrevê-las (as características), através de um estudo de caso.

No protótipo não foi implementado nenhum suporte para se projetar o esquema do banco de dados, mas sim para a manipulação dos objetos e suas versões. Desse modo, foi utilizado um banco de dados exemplo, com seu esquema previamente projetado. O esquema-exemplo utilizado é composto de duas classes, formando uma generalização (*Veículo* ← *Automóvel*). Esse pequeno esquema-exemplo é suficiente para demonstrar as características implementadas no protótipo, para o Browser de Objeto e para o Browser de Consulta.

- *As classes e atributos do esquema-exemplo:*

⇒ *Classe* **Veículo** {
motor : string,
combustível : string,
cor : string }

⇒ *Classe* **Automóvel** *"herança de"* **Veículo** {
volumes : integer,
acessórios : string }

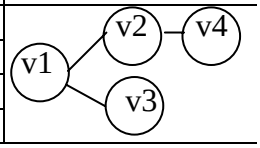
No estudo de caso, serão utilizados alguns objetos não-versionados e versionados (e suas versões) *default*, para demonstrar as características implementadas, principalmente a navegação pelos objetos da classe (como um todo), como pelas versões dos objetos versionados.

As tabelas 5.1 e 5.2 representam um esboço dos objetos *default* para o estudo de caso, onde não foi seguido nenhum formalismo para a representação textual dos objetos não-versionados e dos objetos versionados (e suas versões).

TABELA 5.1- Estudo de Caso - Instâncias no nível de *Veículo*

| **OID** | **Status** | **Motor** | **Combustível** | **Cor** |
|---|---|---|---|---|
| 1 (V) | "não-versionado" | 2.0 mpi | ÁLCOOL | Azul Santiago |
| 2 (V) | "não-versionado" | 1.0 mpi | ÁLCOOL | Cinza Steel |

3 (V) "versionado" (nome do objeto Versionado: FIAT-UNO)

| OID | Versão | Motor | Combustível | Versão Origem | Cor | Grafo |
|---|---|---|---|---|---|---|
| 3-1 | V1 | 1.5 mpi | GASOLINA | - | Azul |  |
| 3-2 | V2 | 1.0 mpi | GASOLINA | V1 | Verde Oliva | |
| 3-3 | V3 | 1.5 mpi | ÁLCOOL | V1 | Cinza Steel | |
| 3-4 | V4 | 1.0 mpi | ÁLCOOL | V2 | Cinza Trend | |



4 (V) "Versionado": nome objeto Versionado: FIAT-PALIO (nível de Veículo)

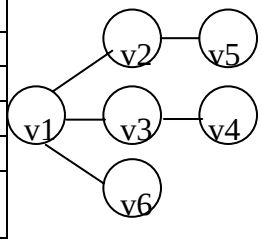
| OID | Versão | Motor | Combustível | Versão Origem | Cor | Grafo |
|---|---|---|---|---|---|---|
| 4-1 | V1 | 1.5 mpi | GASOLINA | - | Bordo |  |
| 4-2 | V2 | 1.0 mpi | GASOLINA | V1 | Cinza Dark | |
| 4-3 | V3 | 1.5 mpi | ÁLCOOL | V1 | Cinza Steel | |
| 4-4 | V4 | 1.0 mpi | ÁLCOOL | V3 | Vermelho | |
| 4-5 | V5 | 1.5 mpi | GASOLINA | V2 | Preto | |
| 4-6 | V6 | 1.0 mpi | ÁLCOOL | V1 | Branco | |



TABELA 5.2- Estudo de Caso - Instâncias no nível de *Automóvel*

| **OID** | **Status** | **Volumes** | **Acessórios** | **OID correspondente em Veículo** |
|---|---|---|---|---|
| 1 (A) | "não-versionado" | 2 | X | 1 (V) |
| 2 (A) | "não-versionado" | 3 | W | 2 (V) |

3 (A) "versionado": nome objeto Versionado: FIAT-UNO (nível Automóvel)

| **OID** | **Versão** | **Volumes** | **Acessórios** | **Versão Origem** | **Versões Correspondentes Em *Veículo*** | **OID correspondente em Veículo** |
|---|---|---|---|---|---|---|
| 3-1(A) | V1 | 2 | X | - | V1, V2 | 3 (V) |
| 3-2(A) | V2 | 2 | Y | V1 | V3, V4 | 3 (V) |
| 3-3(A) | V3 | 2 | Z | V2 | V2, V4 | 3 (V) |
| 3-4(A) | V4 | 2 | W | V2 | V1, V3 | 3 (V) |
| 3-5(A) | V5 | 3 | X | V1 | V1, V2, V4 | 3 (V) |

4 (A) "versionado": nome objeto Versionado: FIAT-UNO (nível Automóvel)

| **OID** | Versão | **Volumes** | **Acessórios** | **Versão Origem** | **Versões Correspondentes Em *Veículo*** | **OID correspondente em Veículo** |
|---|---|---|---|---|---|---|
| 4-1(A) | V1 | 2 | X | - | V1, V3, V4 | 4 (V) |
| 4-2(A) | V2 | 2 | Y | V1 | V3, V5 | 4 (V) |
| 4-3(A) | V3 | 3 | X | V1 | V2, V4, V6 | 4 (V) |
| 4-4(A) | V4 | 3 | W | V1 | V1, V3, V5 | 4 (V) |

**5.4 Suporte para inclusão de objetos (não-versionados)**

Para incluir um objeto (não-versionado), basta pressionar o botão *Novo*, na *barra de status* do browser de objeto (ver item 4.6.3, figura 4.24). Pois, o *default* é incluir primeiro o objeto (não-versionado) para depois, se desejado, versioná-lo (através do botão *Versionar*). Quando é pressionado o botão *Novo*, os demais botões são desabilitados e o botão *Cancelar* (que não estava visível) é habilitado.

A figura 5.6 mostra a inclusão de um objeto, no nível de *Veículo*, sendo esse objeto, um objeto não-versionado. O objeto inserido é representado no exemplo (tabela 5.1) pelo identificador de objeto (OID) de número 1 (um), no nível de *Veículo*: 1 (V).

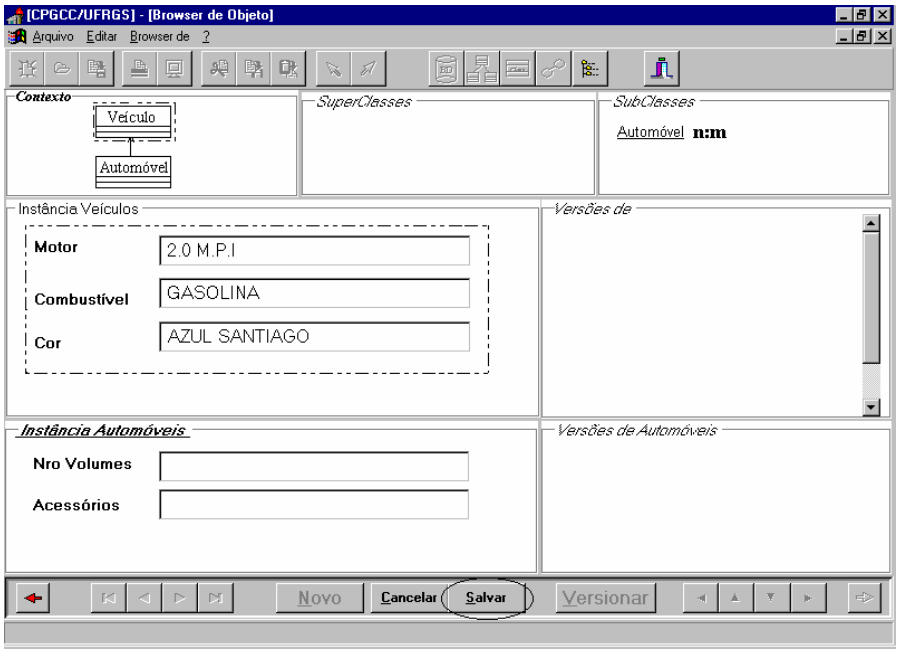



FIGURA 5.6- Incluindo um objeto não-versionado

## 5.5 Suporte para o versionamento de objetos e inclusão de novas versões

Para criar novas versões, deve-se pressionar o botão *Versionar*, na *barra de status* do browser de objeto (ver item 4.6.3, figura 4.24). Caso o objeto *default* for um objeto não-versionado (não possui versões), esse objeto *default* passará a ser a primeira versão do objeto versionado. Caso o objeto *default* for um objeto versionado (possui versões), mais uma versão será criada, usando como versão de origem a versão corrente.

Seguindo o esboço da tabela 5.1, para incluir a versão *v1*, do objeto versionado *fiat-uno*, representada no exemplo pelo OID = “3-1”, pressiona-se o botão *Versionar*, com o objeto “OID=3” ativo (que, no momento, é um objeto não-versionado), e esse passará a ser a primeira versão do objeto versionado *fiat-uno* (ver tabela 5.1). A interface gera, automaticamente, uma cópia da versão corrente (*v1*), e essa cópia (*v2*) passará a ser uma versão derivada da versão corrente (*v1*). A versão *v2* está representada no exemplo (tabela 5.1) pelo OID = “3-2”.

A figura 5.7 mostra o versionamento do objeto de OID = ‘3’. O nome para esse objeto versionado (como mostra o exemplo – tabela 5.1) é *‘FIAT-UNO’*. Esse objeto versionado *fiat-uno* terá duas versões: *v1* e *v2*.

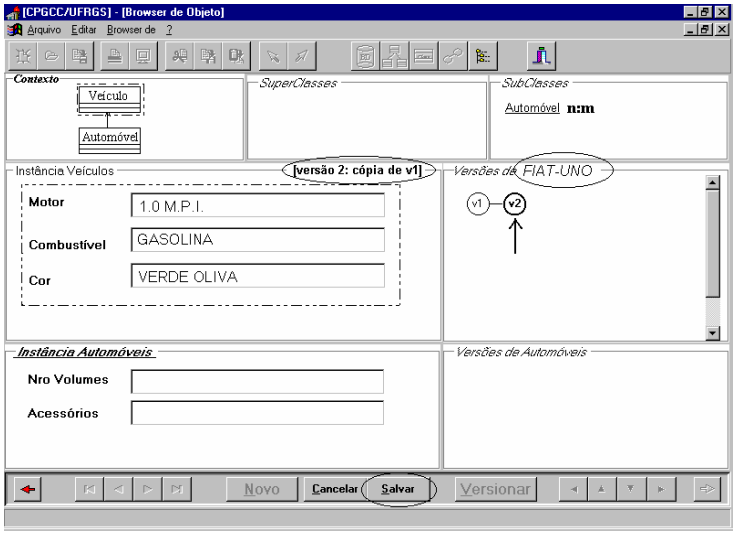



FIGURA 5.7- Versionando um objeto

### 5.5.1 Inclusão de novas versões

Para incluir as demais versões do objeto versionado *fiat-uno*: *v3* e *v4*, procede-se da mesma maneira que o versionamento de objetos, ou seja, pressionando o botão *Versionar*. Um detalhe importante a ser lembrado é que a versão de origem da derivação será a versão corrente. Por isso, seguindo exemplo (tabela 5.1) quando a *v3* for inserida a versão *v1* deverá ser a versão default.

A figura 5.8 mostra a inclusão da versão *v3* do objeto versionado *FIAT-UNO* no nível da classe *Veículo*. A versão *v3* tem como versão de origem a versão *v1*; e a versão *v4*, representada pelo OID = “3-4”, tem como sua versão de origem a versão *v2*.

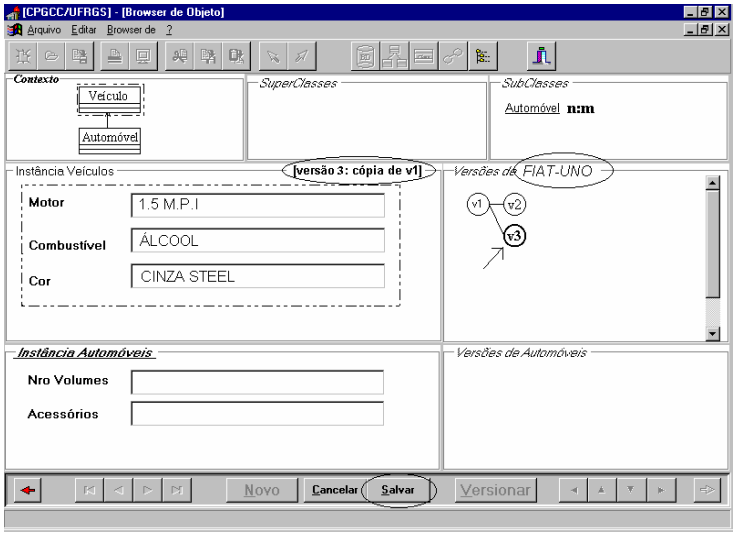



FIGURA 5.8- Inserindo versão *v3* do objeto versionado FIAT-UNO (nível de Veículo)

A inclusão da versão *v4* do objeto versionado *FIAT-UNO,* no nível da classe *Veículo,* segue o mesmo procedimento feito para a inclusão da versão *v3*.

A inclusão do objeto versionado FIAT-PALIO, representado pelo OID = 4 (V), no nível de Veículo, e de suas versões, seguem os mesmos procedimentos descritos acima para a inclusão do objeto versionado FIAT-UNO e suas versões.

## 5.6 Suporte para o versionamento nos vários níveis da hierarquia de classes

Para criar versões nos vários níveis da hierarquia de classes, precisa-se, primeiramente, selecionar a subclasse. Uma subclasse pode ser selecionada (ativada) com um simples clique sobre a subclasse desejada: (a) na *viewport Contexto*; ou (b) na *viewport Instância*; ou (c) na *viewport Subclasses*.

Os passos para criar as versões na subclasse é semelhante às etapas descritas para o versionamento na superclasse. *O diferencial é que na inserção de versões na subclasse, além de escolher quais as versões de origem no mesmo nível de classe, deve-se escolher quais serão as versões correspondentes na superclasse.*

Um aspecto muito importante a ser lembrado é que cada objeto da subclasse *Automóvel* (versionado ou não-versionado) possui um objeto correspondente na superclasse *Veículo*. O objeto correspondente na superclasse é o objeto corrente na *viewport Instância de Veículos*.

Seguindo o exemplo (tabela 5.2), o primeiro objeto da subclasse *Automóvel* é um objeto não-versionado, representado pelo OID = 1 (A). Para incluir esse objeto e os demais objetos da subclasse *Automóvel* deve-se seguir os seguintes passos: (1º) tornar o objeto correspondente no nível da superclasse Veículo, como o objeto corrente; (2º) selecionar a subclasse *Automóvel*; (3º) pressionar o botão *Novo*; (4º) informar os valores para cada atributo dessa subclasse *Automóvel*.

A figura 5.9 mostra a inclusão do primeiro objeto da subclasse *Automóvel*, representado pelo OID = 1 (A). Esse objeto é não-versionado.

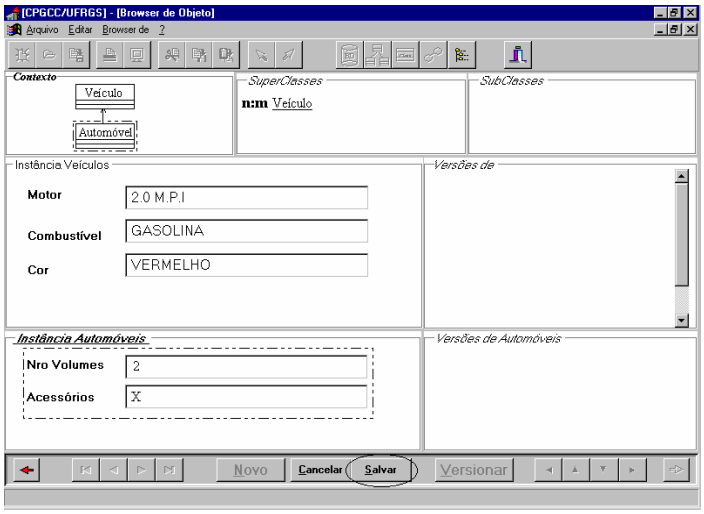



FIGURA 5.9- Inclusão do primeiro objeto na subclasse *Automóvel*

Seguindo o exemplo (tabela 5.2) será inserido o terceiro objeto, representado pelo OID = 3 (A), no nível da subclasse *Automóvel*. Um objeto antes de ser versionado e apresentar versões deve, primeiramente, ser incluído como um objeto não-versionado (através do botão *Novo*). Os seguintes passos se fazem necessários para a inclusão desse objeto: (1º) tornar o objeto versionado *FIAT-UNO* o objeto corrente (ativo) na *viewport Instâncias de Veículo*; (2º) selecionar a subclasse *Automóvel*; e (3º) pressionar o botão *Novo*; (4º) informar os valores para cada atributo dessa subclasse *Automóvel*. O próximo passo é versionar esse objeto, clicando no botão *Versionar*.

*É importante ressaltar que na inserção de versões na subclasse, deve-se escolher quais as versões correspondentes na superclasse, obedecendo a restrição de correspondência: 1:1, n:1, 1:n, n:m. A interface permite ao usuário escolher graficamente as versões que serão as correspondentes na superclasse, controlando a restrição de correspondência automaticamente.*

A figura 5.10 mostra a inclusão do objeto versionado FIAT-UNO no nível da subclasse *Automóvel*, com as suas versões correspondentes na superclasse *Veículo*. Conforme o exemplo (tabela 5.2) a versão *v2* ("*Mille*") tem como versões correspondentes no nível da superclasse *Veículo* as versões *v3* e *v4*. Essas versões *v3* e *v4* foram escolhidas com um simples clique sobre elas na *viewport* versões de FIAT-UNO, no nível de *Veículo*.

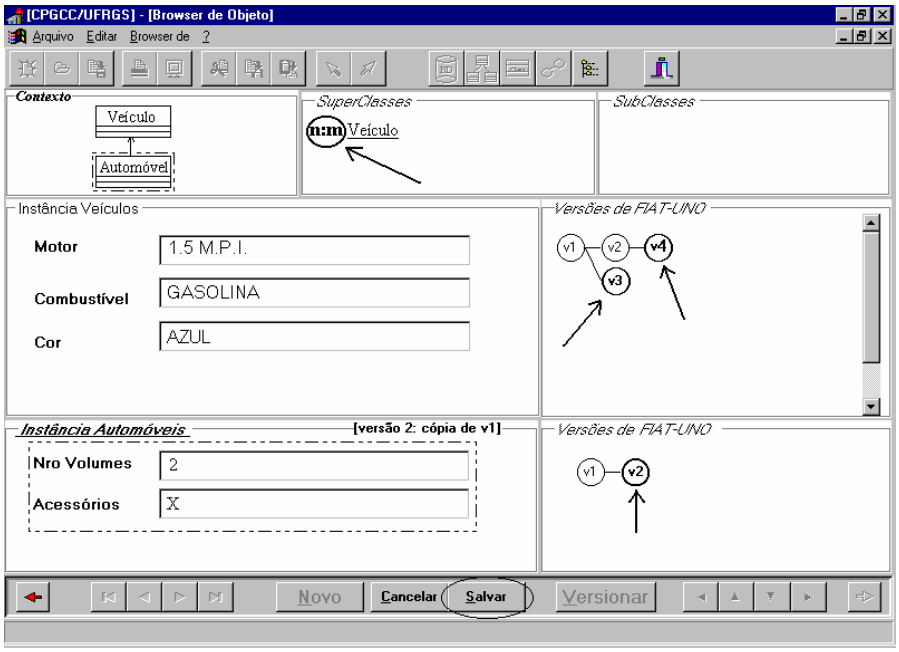



FIGURA 5.10- Inclusão do objeto versionado FIAT-UNO, em Automóvel

No exemplo, a correspondência definida entre a subclasse Automóvel e sua superclasse Veículo foi de 1:n, conforme mostra a *viewport Superclasses. Na correspondência 1:n, uma versão na subclasse pode corresponder a várias versões na superclasse e várias versões na superclasse poderão corresponder a uma versão na subclasse.*

## 5.7 Suporte para a representação gráfica de um objeto versionado

Um objeto versionado é representado por um *Grafo de derivação* que apresenta as versões graficamente, através de círculos. Esse grafo de derivação é visualizado na viewport grafo (Versões de ...). Algumas considerações sobre o grafo devem ser mencionadas:

- O grafo mostra graficamente todas as versões do objeto versionado ativo. A viewport grafo apresenta uma *barra de scroll* no caso de não ser possível visualizar todas as versões;
- As versões correntes aparecem no grafo, destacadas **em negrito** e os seus atributos são visualizados nas *viewport Instâncias*, tanto no nível de superclasse (*Veículo*) como a nível de subclasse (*Automóvel*);
- Quando a superclasse está destacada, as versões correspondentes na subclasse, da versão corrente, também aparecem destacadas, mas com o contorno do círculo (que representa a versão), pontilhado;
- Quando a subclasse está destacada, as versões correspondentes na superclasse, da versão corrente, também aparecem destacadas, mas com o contorno do círculo (que representa a versão), pontilhado;

A figura 5.11 mostra a representação gráfica do objeto versionado *FIAT-UNO*, com a *v3* como a versão corrente no nível de *Veículo*. As versões *v2* e *v4* aparecem destacadas no nível de *Automóvel* (pois se correspondem com a versão v3 em *Veículo*), sendo *v2* a versão corrente no nível de Veículo.

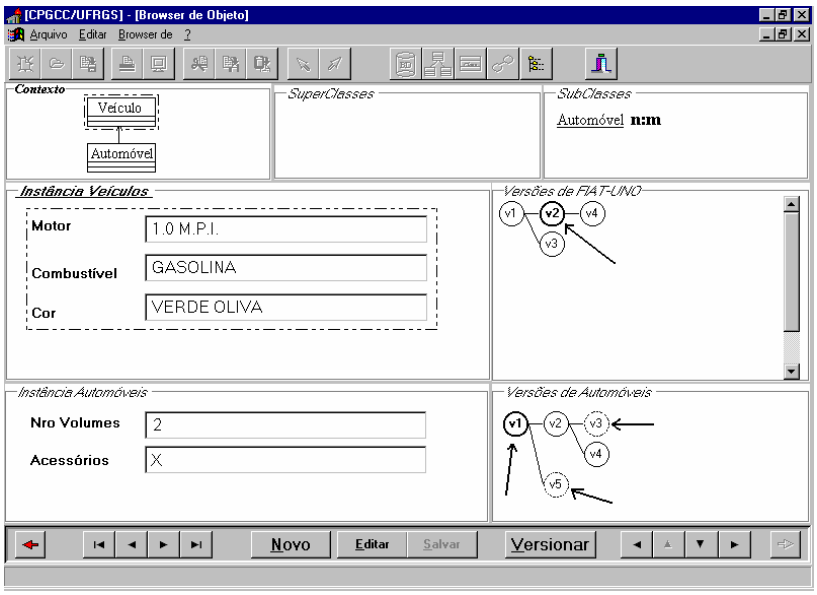



FIGURA 5.11- Representando graficamente o objeto versionado *FIAT-UNO*

Na Figura 5.11, a versão corrente no nível da superclasse *Veículo* é a versão *v2*, e a versão corrente no nível da subclasse *Automóvel* é a versão *v1*, onde ambas aparecem

destacadas ***em negrito*** e os valores de seus atributos aparecem nas *viewports Instâncias*. É importante ressaltar que as versões correspondentes aparecem destacadas através do contorno pontilhado e somente são visualizadas as versões correspondentes do nível de classe que não está em destaque, ou seja, no exemplo a classe em destaque é a superclasse *Veículo*. Então, as versões correspondentes a serem destacadas (contorno pontilhado) serão as versões da subclasse *Automóvel* que se correspondem com a *v2* em *Veículo*, que no exemplo são as versões *v1, v3,* e *v5* em *Automóvel*.

Para o melhor entendimento da representação gráfica de um objeto versionado, versões em destaque e as versões correspondentes, nos vários níveis da hierarquia de classes, serão organizadas duas figuras, simulando a representação gráfica do objeto versionado *FIAT-UNO* (OID = 3): (a) uma no nível de *Veículo;* e (b) outra no nível de *Automóvel*. Também serão apresentadas todas as combinações de versões correntes e versões correspondentes (da versão corrente) descritas no estudo de caso (tabela 5.1 e 5.2) para esse objeto versionado, representado pelo OID = 3. Por questões de espaço nas figuras, serão mostradas apenas os *grafos de derivação*.

A figura 5.12 mostra uma simulação de todas as possíveis representações gráficas do objeto versionado FIAT-UNO, no nível da superclasse *Veículo*, com as versões correntes em *Veículo* e as versões correspondentes em *Automóvel*.

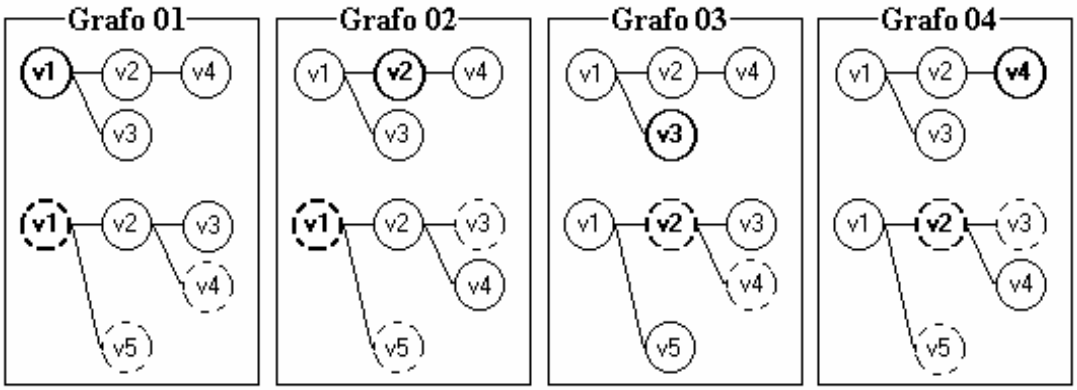



FIGURA 5.12- *Grafos de Derivação* do objeto versionado FIAT-UNO, nível de *Veículo*

O grafo 01, da figura 5.12, representa o objeto versionado FIAT-UNO, com a versão ***v1*** como a versão corrente, no nível de *Veículo*. As versões *v1, v4, v5* em *Automóvel* se correspondem com a versão *v1* em *Veículo* e por isso aparecem em destaque, sendo das três versões (*v1, v4, v5*) em *Automóvel,* a versão ***v1*** é a versão corrente.

No grafo 02, da figura 5.12, a versão ***v2*** é a versão corrente, no nível de *Veículo*. Das três versões (*v1, v3, v5*) correspondentes em *Automóvel,* a versão ***v1*** é a versão corrente.

No grafo 03, da figura 5.12, a versão ***v3*** é a versão corrente, no nível de *Veículo*. Das duas versões (*v2, v4*) correspondentes em *Automóvel,* a versão ***v2*** é a versão corrente.

No grafo 04, da figura 5.12, a versão ***v4*** é a versão corrente, no nível de *Veículo*. Das três versões (*v2, v3, v5*) em *Automóvel,* a versão ***v2*** é a versão corrente.

A figura 5.13 mostra uma simulação de todas as possíveis representações gráficas do objeto versionado FIAT-UNO, no nível da subclasse *Automóvel*, com as versões correntes em *Automóvel* e as versões correspondentes em *Veículo*.

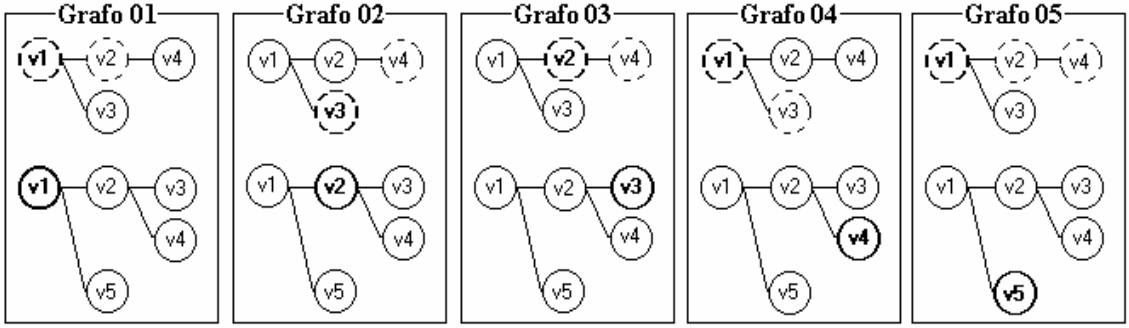



FIGURA 5.13- *Grafos de Derivação* do objeto versionado FIAT-UNO, nível de *Automóvel*

O grafo 01, da figura 5.13, representa o objeto versionado FIAT-UNO, com a versão ***v1*** como a versão corrente, no nível da subclasse *Automóvel*. As versões *v1, v2* em *Veículo* são as versões correspondentes da versão *v1* em *Automóvel* e por isso aparecem em destaque, sendo, das duas versões (*v1, v2*) em *Automóvel,* a versão ***v1*** a versão corrente.

No grafo 02, da figura 5.13, a versão ***v2*** é a versão corrente, no nível da subclasse *Automóvel*. Das duas versões (*v3, v4*) correspondentes em *Veículo,* a versão ***v3*** é a versão corrente.

No grafo 03, da figura 5.13, a versão ***v3*** é a versão corrente, no nível da subclasse *Automóvel*. Das duas versões (*v2, v4*) correspondentes em *Veículo,* a versão ***v2*** é a versão corrente.

No grafo 04, da figura 5.13, a versão ***v4*** é a versão corrente, no nível da subclasse *Automóvel*. Das duas versões (*v1, v3*) correspondentes em *Veículo,* a versão ***v1*** é a versão corrente.

No grafo 05, da figura 5.13, a versão ***v5*** é a versão corrente, no nível da subclasse *Automóvel*. Das três versões (v1, *v3, v5*) correspondentes em *Veículo,* a versão ***v1*** é a versão corrente.

## 5.8 Suporte para a navegação pelos objetos e versões

Quando o browser de objeto é ativado, por *default*, é selecionado o primeiro objeto (versionado ou não-versionado) da classe ativa. A partir desse objeto é que se inicia a navegação pelos objetos da classe (versionados ou não-versionados) ou pelas versões dos objetos versionados. Para percorrer os objetos versionados ou não-versionados da classe como um todo, utiliza-se o primeiro conjunto de botões, localizado à esquerda da *barra de status* do browser de objetos (ver item 4.6.3, figura 4.24). Para navegar pelas versões de um objeto versionado, utiliza-se o segundo conjunto de botões para navegação, à direita da *barra de status* do browser de objeto. Uma descrição das funcionalidades de cada botão de navegação aparece no item 4.6.3.

### 5.8.1 Suporte para a navegação pelas versões de um objeto versionado

Para melhor entendimento da navegação pelas versões de um objeto versionado será utilizado o objeto versionado FIAT-PALIO, representado pelo OID = 4, para uma simulação de navegação pelas suas versões. Essa simulação será representada pelas figura 5.14 e 5.15 (a figura 5.15 é continuação da figura 5.14).

Antes de visualizar as simulações de navegação pelas versões de FIAT-PALIO algumas considerações devem ser descritas.

- O usuário pode navegar pelas versões de um objeto versionado de várias maneiras:
  - utilizando o mouse: através de um simples clique sobre a versão desejada (no grafo de derivação);
  - utilizando a barra de navegação, formada por um conjunto de botões. Essa barra é o segundo conjunto de botões, localizado à direita da *barra de status* do browser de objeto (ver figura 4.24);
  - utilizando as setas: $\leftarrow$, $\uparrow$, $\downarrow$, $\rightarrow$;
  - os critérios de navegação pelas versões estão descritos juntamente com o exemplo da figura 4.25;

- as ações dos botões de navegação ou das setas ($\leftarrow$, $\uparrow$, $\downarrow$, $\rightarrow$) terão efeito sobre a classe que está em destaque. No exemplo, se a superclasse *Veículo* estiver em destaque, as ações dos botões afetarão o grafo de derivação da versão *Veículo*.

- A forma de navegação pelas versões de um objeto versionado no nível de superclasse se aplica da mesma maneira para a navegação pelas versões de um objeto versionado no nível de subclasse.

- É importante ressaltar que a cada movimento (mudança de uma versão para outra), tanto no nível de superclasse, como a nível de subclasses, as *viewports Grafos* e *viewports Instâncias* serão atualizadas com as novas informações, ou seja, será destacada a nova versão corrente e as suas versões correspondentes.

A figura 5.14 mostra uma simulação de navegação pelas versões do objeto versionado FIAT-PALIO. A simulação percorrerá o caminho v1 - v2 - v3 - v4 - v5 - v6, no nível da superclasse *Veículo*. Vários outros caminhos poderiam ser simulados.

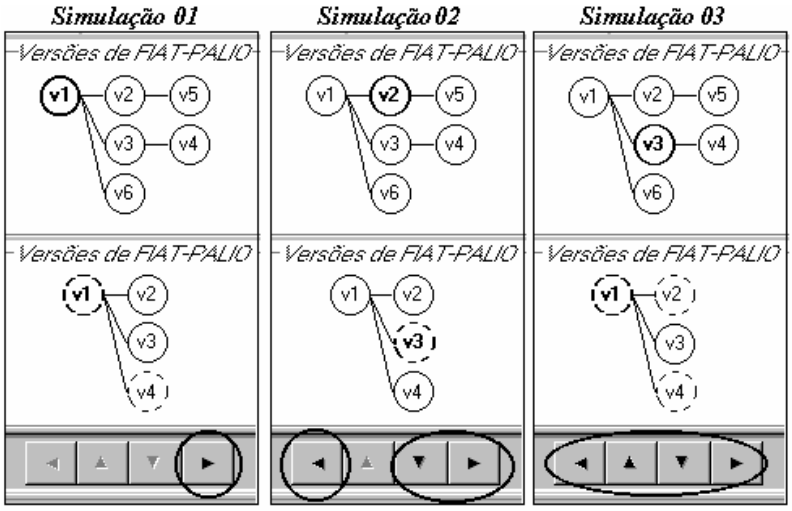



FIGURA 5.14- Navegando pelas versões de FIAT-PALIO – simulações I

Na *simulação 01,* a versão *v1,* em *Veículo,* é a versão corrente. E os movimentos possíveis de navegação são representados pelos botões de navegação. No exemplo a única navegação possível é ir para a próxima versão (→), ou seja, ir para a versão *v2*. Isso no nível da superclasse *Veículo*. É importante ressaltar que a utilização do mouse modifica essa ordem de navegação, ou seja, o usuário pode clicar em qualquer versão do grafo de derivação.

Na *simulação 02,* a versão *v2,* em *Veículo,* é a versão corrente. No exemplo as navegações possíveis são: (a) voltar para a versão anterior (←), que é a versão *v1*; (b) descer para a versão abaixo horizontalmente (↓), que é a versão *v3*; (c) ir para a próxima versão (→), que é a versão *v5*. Isso tudo no nível da superclasse *Veículo*. No exemplo, optou-se por navegar para a versão *v3* (abaixo horizontalmente).

Na *simulação 03*, a versão *v3*, em *Veículo,* é a versão corrente. No exemplo, todas as navegações são possíveis: (a) voltar para a versão anterior (←), que é a versão *v1*; (b) subir uma versão acima horizontalmente ( ↑ ), que é a versão *v2*; (c) descer uma versão abaixo horizontalmente (↓), que é a versão *v6*; (d) ir para a próxima versão (→), que é a versão *v4.* Isso tudo no nível da superclasse *Veículo*. No exemplo, optou-se por navegar para a versão *v4* (próxima versão).

A figura 5.15 mostra a continuação das simulações de navegação pelas versões do objeto versionado FIAT-PALIO.

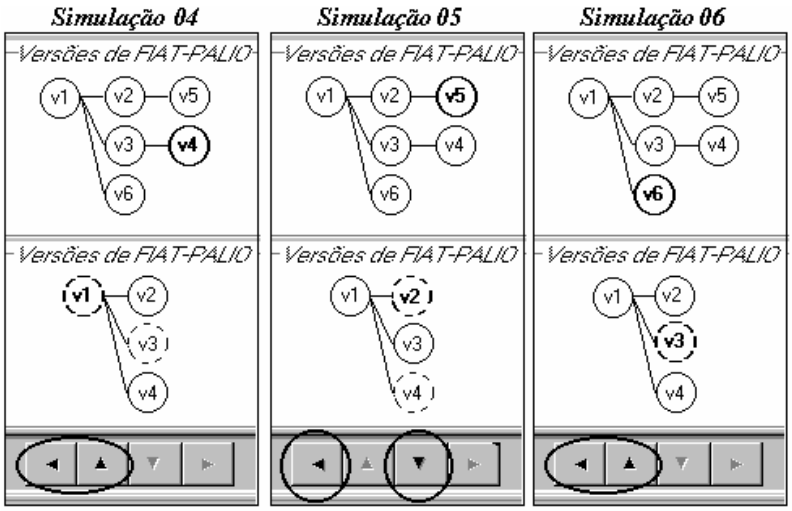



FIGURA 5.15- Navegando pelas versões de FIAT-PALIO – simulações II

Na *simulação 04,* a versão *v4,* em *Veículo,* é a versão corrente. No exemplo, as navegações possíveis são: (a) voltar para a versão anterior (←), que é a versão *v3*; (b) subir uma versão acima horizontalmente ( ↑ ), que é a versão *v5*. Isso tudo no nível da superclasse *Veículo*. No exemplo optou-se por navegar para a versão acima horizontalmente, que é a versão *v5*.

Na *simulação 05,* a versão *v5,* em *Veículo,* é a versão corrente. No exemplo as navegações possíveis são: (a) voltar para a versão anterior (←), que é a versão *v2*; (b) descer uma versão abaixo horizontalmente (↓), que é a versão *v4*. Isso tudo no nível da superclasse *Veículo*. No exemplo optou-se por navegar para a versão v6, que foi selecionada com o mouse (outro caminho seria: voltar para *v2*, descer para *v3* e enfim descer para *v6*).

Na *simulação 06,* a versão *v6,* em *Veículo,* é a versão corrente. No exemplo as navegações possíveis são: (a) voltar para a versão anterior (←), que é a versão *v1*; (b) subir uma versão acima horizontalmente ( ↑ ), que é a versão *v3*. Isso tudo no nível da superclasse *Veículo*.

Observou-se que a implementação de um protótipo da interface, com algumas das características descritas à nível de projeto (protótipo do Browser de Objetos e do Browser de Consulta) se justifica pelos seguintes aspectos, entre outros:

- muitas características previstas à nível de projeto, podem não ser passíveis de implementações práticas, sem uma verificação prévia através de um protótipo;
- serve de orientação para propor novas caraterísticas para a interface;
- é útil para reforçar o entendimento das caraterísticas, propostas, à nível de projeto.

# 6 Conclusões

Este trabalho apresentou o projeto de uma interface visual para MDOOs, com suporte para o modelo de versões, descrito em [GOL 95]. Foi implementado um protótipo com algumas das características projetadas para a interface, especialmente as propostas para o browser de objeto e o seu suporte para versões. No projeto da interface, preocupou-se em acrescentar funcionalidades de interfaces existentes para modelos de dados orientados a objetos.

O projeto de uma interface visual para MDOOs, com suporte para versões de objetos, não é uma tarefa simples, pois vários aspectos devem ser analisados, entre outros:

- a abertura ou criação de um banco de dados de objetos do SGBDOO $O_2$ ou de outro SGBDOO;
- o projeto do esquema, classes e relacionamentos de um BDO;
- a manipulação (visualização e atualização) dos objetos e, conseqüentemente, das suas versões;
- a representação gráfica das informações (esquema, objetos, versões..);
- a consulta aos dados armazenados;
- a capacidade de definir e visualizar versões de um objeto nos vários níveis da hierarquia de classes;
- definir e visualizar as correspondências entre as versões. As correspondências podem ser estáticas ou dinâmicas (correspondência a objetos compostos ou versionados - configurações de versões);

Ao projeto da interface, procurou-se incorporar características de outras interfaces, sejam experimentais ou comerciais, para modelos de dados orientados a objetos. Observou-se a importância do estudo de outras interfaces, pois implementações e enfoques diferenciados aumentam o leque de possibilidades para se projetarem boas funcionalidades na interface. Também buscou-se, no projeto da interface, utilizar algum formalismo para representar graficamente as informações a serem visualizadas pelo usuário. Por exemplo, como representar graficamente o esquema de um banco de dados? Como representar as suas classes, atributos, métodos e relacionamentos?

Observou-se que é mais intuitivo apresentar as informações do banco de dados ao usuário, agrupando-as por características semelhantes. Para isto, surge a necessidade de se estabelecerem critérios de afinidades. Sugere-se apresentar as informações de um banco de dados através de browsers específicos, pois cada um trata de aspectos particulares da informação global. Os browsers sugeridos são:

- *browser de banco de dados:* para as características relacionadas com a abertura e a criação do banco de dados;

- *browser de esquema:* para as características envolvidas com o projeto do esquema do banco de dados;

- *browser de classe:* para as características que dizem respeito a uma classe individualmente, com seus métodos e atributos;

- *browser de objeto:* para as características relacionadas com a manipulação dos objetos e suas versões;

- *browser de consulta:* para as características que envolvem consultas aos objetos do banco de dados.

A navegação é recomendada para a investigação dos objetos de uma classe ou das versões de um objeto versionado, pois é mais intuitivo visualizar os objetos, percorrendo-os "um-a-um". Um browser visual se faz necessário para a realização da navegação e também para a realização de consultas aos objetos versionados e suas diferentes versões, pois torna-se mais natural navegar e consultar através de uma representação gráfica. Observou-se que a consulta visual necessita de um estudo mais aprofundado, pois consultas mais sofisticadas requerem a utilização de predicados (>, <, =, >=, <=, AND, OR). Esse estudo se faz necessário para não tornar a interface complexa e de difícil utilização. Consultas simples, onde não é necessário o conhecimento prévio do sistema, ficam mais fáceis de serem formuladas a partir da utilização de mecanismos visuais; enquanto consultas mais elaboradas, que exigem um maior nível de conhecimento do sistema pelo usuário, necessitam de uma linguagem textual.

No protótipo (browser de objeto) não foi implementado um aspecto importante, que diz respeito ao modelo de versões: o suporte à gerência de configurações para modelos de dados orientados a objetos com suporte para a representação de versões. Observa-se que esse aspecto é de grande importância e merece considerável atenção.

Uma proposta futura é incorporar a interface o suporte à gerência de configurações, pesquisa que está sendo desenvolvida atualmente no CPGCC [SCH 96]. Tal trabalho visa identificar, especificar e implementar os requisitos necessários à gerência de configurações para o MDOOs com suporte para a representação de versões apresentado em [GOL 95]. Tal trabalho também apresenta uma proposta de linguagem para a gerência de configurações, além de discutir a especificação de uma interface gráfica, que conforme o autor, serviria como ferramenta de apoio na gerência de configurações [SCH 96].

Portanto, espera-se que este trabalho de dissertação contribua e incentive o desenvolvimento de novas pesquisas sobre interfaces visuais para banco de dados orientados a objetos e versões, e também desperte novas idéias para o projeto de funcionalidades, envolvendo a interação "Usuário-BDO", em especial a navegação.

# Bibliografia

[AC 93] APPLE COMPUTER, Inc. **Macintosh Human Interface Guidelines.** Reading, MA: Addinson-Wesley, 1992.

[AGR 90] AGRAWAL, R.; GEHANI, N. H.; SRINIVASAN, J. OdeView: The Graphical Interface to Ode**. SIGMOD Record**, New York, v.9, n.2, June 1990. Trabalho apresentado na ACM SIGMOD International Conference on Management of Data, 1990.

[ATK 89] ATKINSON, M. et al. The Object-Oriented Database System Manifesto. In: INTERNATIONAL CONFERENCE ON DEDUCTIVE AND OBJECT-ORIENTED DATABASES, 1., 1989. **Proceedings** … Kyoto, Japan: [s.n.], 1989. p. 40-57.

[BAS 93] BASTIEN, Christian; SCAPIN, Dominique. **Ergonomic criteria for the evaluation of human-computer interface**. [S.l.: s.n.], 1993. (Rapport technique, n.156).

[BER 93] BERTINO, Elisa; MARTINO, Lorenzo. **Object-Oriented Database System:** concepts and architectures**.** Workingham: Addison-Wesley, 1993.

[BIL 90] BILIRIS, A. Modeling design object relationships in PEGASUS. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 6., 1990, Los Angeles, USA. **Proceedings...** Los Alamitos: IEEE, 1990. p.228-236.

[BOE 88] BOEHM, B.W. A Spiral Model of Software Development and Enhancement. **Computer**, New York, v.21, n.5, p.61-72, May 1988.

[BOO 91] BOOCH, Grady. **Object-Oriented Design. With Aplications**. Redwood: The Benjamin/Cummings, 1991. 580p.

[BUT 91] BUTTERWORTH, Paul; OTIS, Allen; STEIN, Jacob. The Gemstone Object Database Management System**. Communications of the ACM**, New York, v.34, n.10, p.64-77, Oct. 1991.

[CAR 95] CAREY, Michael *et al*. Towards Heterogeneous Multimedia Information Systems: the Garlic Approach. In: INTERNATIONAL WORKSHOP ON RESEARCH ISSUES IN DATA ENGINEERING: DISTRIBUTED OBJECT MANAGEMENT, 5., 1995, Taiwan. **Proceedings...** [S.l.: s.n.], 1995.

[CAR 96] CAREY, Michael *et al*. PESTO: An Integrated Query/Browser for Object Databases. In: VLDB CONFERENCE, 22., 1996, Mubai, India. **Proceedings …** Disponível por WWW em http://www.almaden.ibm.com/cs/garlic/pesto_vldb96.

[CAT 96] CATTELL, Rick and et al. **The Object Database Standard: ODMG-93** (Release 1.2). San Francisco, CA: Morgan Kaufmann, 1996.

[CHA 96] CHAN, Silvio; ROCHA, Heloísa Vieira da Rocha; DRUMMOND, Rogério. **Métodos e Parâmetros para Avaliação de Interfaces Homem-Computador.** Campinas: DCC - IMECC - UNICAMP, 1996. Proposta de dissertação de mestrado.

[COA 91] COAD, Peter; YOURDAN, Edward. **Object-Oriented Analysis**. 2.ed. Englewood Cliffs: Prentice-Hall, 1991. 233p.

[DEU 91] DEUX, O. et al. The $O_2$ System. **Communications of the ACM**, New York, v.34, n.10, p.35-48, Oct. 1991.

[DOW 91] DOWNTON, Andy. **Engineering the Human-Computer Interface.** [S.l.]: McGraw-Hill, 1991.

[GAL 97] GALANTE, Renata de Mattos. **Evolução de Esquemas em banco de dados orientados a objetos com suporte para o modelo de versões**: (relatório de pesquisa). Porto Alegre: CPGCC/UFRGS, 1996.

[GOL 95] GOLENDZINER, Lia Goldstein. **Um modelo de versões para banco de dados orientados a objetos**. Porto Alegre: CPGCC/UFRGS, 1995. Tese de Doutorado.

[GRU 93] GRUNDIN, Jonathan. Interface an Evolving Concept**. Communications of the ACM**, New York, v.26, n.4, p.110-119, Apr. 1993.

[HAR 87] HAREL, D. Statecharts: On the formal Semantics of Statecharts. In: IEEE SYMPOSIUM ON LOGIC IN COMPUTER SCIENCE, 2., 1987, Ithaca. **Proceedings ...** [S.l.: s.n.], 1987.

[IBM 92] IBM. **Object-Oriented Interface Design: IBM Common User Acces Guidelines.** Carmel, Indiana: Que, 1992. 708 p.

[JEF 91] JEFFRIES, Robin *et al*. User Interface evaluation in the real word: a comparison of four techniques. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI, 1991. **Proceedings ...** [S.l.: s.n.], 1991.

[KIN 96] KING, Margaret. Evaluting Natural Language Processing**. Communications of the ACM**, New York, v. 39, n. 1, p.73-79, Jan. 1996.

[KOR 93] KORTH, Henry F.; SILBERSCHATZ, Abraham. **Sistemas de banco de dados**. 2.ed. São Paulo: Makron Books, 1993. 748p. p.18-19.

[LAB 97] LABUTIL. **Critérios Ergonômicos para Avaliação de Interfaces Homem-Computador.** Florianópolis-SC, Labutil: Laboratório de Utilizabilidade, 1997. Disponível por WWW em http://www.ctai.rct-sc.br/labiutil/.

[LEV 94] LEVY, Jonathan. Measuring Usability: Preference vs. Performance**. Communications of the ACM**, New York, v.37, n.4, p.66-75, Apr. 1994.

[LEW 96] LEWIS, David D.; JONES, Karen Spark. Natural Language Processing for Information Retrievel**. Communications of the ACM**, New York, v.39, n.1, p.93-101, Jan. 1996.

[LUC 97a] LUCENA, Fábio Nogueira de; LIESENBERG, Hans K.E. **Bibliografia Contextualizada sobre Interfaces Homem-Computador.** Projeto Xchart. Campinas/SP: DCC/IMECC/UNICAMP, 1997. Disponível por www em http://dcc.unicamp.br/proj-xchart.

[LUC 97b] LUCENA, Fábio Nogueira de. LIESENBERG, Hans K.E. **Fundamentos de Interfaces Homem-Computador.** Campinas: DCC/IMECC/UNICAMP, 1997. Disponível por www em http://dcc.unicamp.br/proj-xchart.

[MAT 95] MATIAS, Márcio; CYBIS, Walter de Abreu. **CHECKLIST: Uma Ferramenta de suporte à avaliação ergonômica de interfaces**. Florianópolis: Programa de Pós-Graduação em Engenharia de Produção - UFSC, 1995. Dissertação de mestrado.

[MEL 94] MELLO, Ronaldo dos Santos. **Uma Linguagem Visual de Consulta para o Ambiente STAR**. Porto Alegre: CPGCC/UFRGS, 1994. Dissertação de mestrado.

[MEY 96] MEYER, Brad A. **A Brief History of Human Computer Interaction Technology.** [S.l.] Carnegie Mellon Unversity, 1996 (technical report CMU-CS-96-163). Human Computer Interaction Institute (technical report CMU-HCII-96-103). Disponível por www wm http://www.cs.cmu.edu/~amulet/papers/.

[MS 97] MICROSOFT CORPORATION (Tandy Trower). **The Windows Interface Guidelines for Software Design.** Redmond, WA: Microsoft Press, 1997.

[NIE 92] NIELSON, Jakob. The Usability Engineering Life Cycle. **Computer**, New York, v.26, n.11, p.12-22, Mar. 1992.

[NIE 93] NIELSON, Jakob. Interactive User-Interface Design**. Computer**, New York, v.26, n.11, p.32-41, Nov. 1993.

[O2 95] $O_2$ TECHNOLOGY. **$O_2$ System**: Administration guide: Release 4.6. Versalles: $O_2$ Technology, 1996.

[O2 96] $O_2$ TECHNOLOGY. **$O_2$ Version**. The $O_2$ Manager. January 1996. Disponível por www em http://www.o2tech.fr/.

[OLI 93a] OLIVEIRA, Juliano Lopes de; ANIDO, Ricardo de Oliveira. **Uma ferramenta gráfica para navegação e consulta em banco de dados orientados a objetos**. Campinas: Departamento de Ciência da Computação - UNICAMP, 1993. Dissertação de mestrado.

[OLI 93b] OLIVEIRA, Juliano Lopes de.; ANIDO, Ricardo de Oliveira. Navegação e Consulta em Banco de Dados Orientados a Objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 8., 1993, Campina Grande. **Anais ...** Disponível www em http://www.dcc.unicamp.br/~juliano.

[OLI 93c] OLIVEIRA, Juliano Lopes de.; ANIDO, Ricardo de Oliveira. Integração de uma Interface para navegação em diferentes sistemas de banco de dados orientados a objetos. In: BRAZILIAN COMPUTER SOCIETY CONGRESS, 13., 1993. **Proceedings ...** [S.l.: s.n.], 1993. p.61-75. Disponível por www em http://www.dcc.unicamp.br/~juliano.

[OLI 93d] OLIVEIRA, Juliano Lopes de; ANIDO, Ricardo de Oliveira. Browsing and Querying in Object-Oriented Databases**.** In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, CIKM', 2., 1993. **Proceedings ...** [S.l.: s.n.], 1993. Disponível por www em http://www.dcc.unicamp.br/~juliano.

[OLI 94] OLIVEIRA, Juliano Lopes de.; ANIDO, Ricardo de Oliveira. On the development of User Interface Systems for Object-Oriented Databases. In: ACM WORKSHOP ON ADVANCED VISUAL INTERFACES, AVI, 1994. **Proceedings ...** [S.l.: s.n.], 1994. p.237-239. Disponível por www em http://www.dcc.unicamp.br/~juliano.

[PER 89] PERLMAN, Gary. **User Interface Development.** [S.l.]: Software Engineering Institute, Carnigie Mellon University, 1989. (Technical Report SEI-CM-17-1.1)

[POW 90] POWEL, James E. **Designing user Interfaces.** San Marcos: Microtend, 1990.

[PRE 94] PREECE, Jenny et al. **Human-Computer Interaction.** [S.l.]: Addison-Wesley, 1994.

[RAT 97a] RATIONAL SOFTWARE CORPORATION. **UML: Unified Modeling Language (Version 1.0).** Santa Clara: Rational Software Corporation, 1997. 3v. As mais recentes atualizações desse documento estão disponíveis por www em http://www.rational.com.

[RAT 97b] RATIONAL SOFTWARE CORPORATION. **Rational Rose/C++ (Version 4.0.3).** A versão de demonstração dessa ferramenta CASE está disponível por www em http://www.rational.com.

[ROT 96] ROTH, M.T. et al. The Garlic project**.** In Proc. Of the **ACM SIGMOD**, New York, v.25, n.2, p.557, June 1996. Trabalho apresentado na ACM SIGMOD International Conference on Management Data, 1996, Montreal, CA.

[RUM 91] RUMBAUGH, James et al. **Object-Oriented Modeling and Design**. Englewood Cliffs: Prentice-Hall, 1991. 431p.

[SAN 86] SANTOS, Clésio Saraiva dos; OLIVEIRA, José Palazzo Moreira de; CASTILHO, José Mauro Volkmer de. O modelo E. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 13.,19-25 julho 1986, Olinda. **Anais...** Recife: SBC/UFPE, 1986. p.342-350.

[SAN 89] SANTOS, Clésio Saraiva dos. **O modelo E revisado**. Porto Alegre: CPGCC/UFRGS, 1989. (Relatório de pesquisa, 119).

[SCH 96] SCHRAMM, Mauro. **Gerência de Configurações em banco de Dados Orientado a Objetos**. Porto Alegre: CPGCC/UFRGS, 1996. Relatório de pesquisa.

[SCL 94] SCLOVSKY, Luciano. **Uma Análise Introdutória da Realidade Virtual.** Porto Alegre: CPGCC/UFRGS, 1994. Dissertação de mestrado.

[SHN 83] SHNEIDERMAN, Ben. Direct Manipulation: A Step Beyond Programming Languages. **IEEE Computer**, New York, v.16, n.8, p.57-69, Aug. 1983.

[SHN 96] SHNEIDERMAN, Ben. **The Eyes Have It**: A Task by Data Type Taxonomy for Information Visualizations**.** Maryland: University of Maryland, College Park, Maryland, 1996. Disponível por www em http://www/cs.umd.edu/projects/hcil/.

[SIL 96] SILVA, Juliano Tonezer da. **Um Estudo de Interfaces Visuais e seus Estilos de Interação para Bancos de Dados Orientados a Objetos com suporte para versões**: Trabalho Individual. Porto Alegre: CPGCC/UFRGS, 1996.

[SJO 96] SJOERD, Michels. **Co-writing, Look and Feel**. Tilburg, The Netherlands: Katholieke Universiteit Brabant, 1996. Master Thesis. Disponível por www em http://infolabwww.kub.nl:2080/w3thesis/.

[SOL 92] SOLOVIEV, Valery. An Overview of Three commercial Object-Oriented Database Systems. ONTOS, ObjectStore, and $O_2$. **SIGMOD RECORD**, New York, v.21, n.1, p.93-104, Mar. 1992.

[SUN 90] SUN MICROSYSTEMS. **OPENLOOK - Graphical User Interface Applications Style Guidelines**. [S.l.]: Addison-Wesley, June 1990.

[VIV 96] VIVI SOFTWARE SRL. **Objecstore Inspector: The Visual Browser for ObjectStore.** 1995-1996. Disponível por www em http://www.dsnet.it/vivi/vom/.

[WIE 94] WIENER, Janet L; NAUGHTON, Jeffrey F. Bulk Loading into an OODB: A Performance Study**.** In: VLDB CONFERENCE, 20., 1994, Santiago, Chile. **Proceedings ...** [S.l.: s.n.], 1994.