UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
DEPARTAMENTO DE INFORMATICA
CURSO DE POS-GRADUAÇÃO EM CIENCIA DA COMPUTAÇÃO

L A G O - Linguagem de Acesso Global ao Sistema AMPLO

por

Paulo Roberto Gomes Luzzardi

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

Prof Dr Flávio Rech Wagner
Orientador

Profa Msc Carla Maria Dal Sasso-Freitas
Co-orientadora

Porto Alegre, novembro de 1991

Luzzardi, Paulo Roberto Gomes

LAGO: Linguagem de Acesso Global ao sistema AMPLO / Paulo Roberto Gomes Luzzardi. - Porto Alegre: CPGCC da UFRGS, 1991.

147p.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul, curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1991. Orientador: Wagner, Flávio Rech. Coorientador: Freitas, Carla Maria Dal Sasso.

Dissertação: projeto auxiliado por computador, interface com usuário, gerência de projeto, ferramentas de projeto, base de dados de projeto

AGRADECIMENTOS

Agradeço o auxílio, a paciência, o incentivo e a sabedoria dos orientadores CARLA M. DAL SASSO FREITAS e FLAVIO RECH WAGNER.

Aos meus pais GILBERTO CECILIANO LUZZARDI e IRLANI GOMES LUZZARDI pelos ensinamentos.

Em especial a minha esposa ADRIANE MARIA MACHADO DE FREITAS LUZZARDI, minha grande inspiração.

Ao meus irmãos FRANCISCO, RENATO, JULIA e ANA pelo incentivo de continuar.

Ao meu grande amigo RICARDO ANDRADE CAVA pela sabedoria, ajuda e incentivo nos piores momentos.

Aos meus amigos PAULO RECH WAGNER, ANA TEREZA, GLAUCIUS, JULIO, JOSE CARLOS, ROBERT, ERICO e os demais ("futebol e tênis") que colaboram com suas idéias, críticas, companheirismo e conhecimentos na elaboração do meu trabalho.

A Universidade Federal do Rio Grande do Sul pelos recursos computacionais, e a CAPES e CNPQ pelo apoio financeiro no projeto de pesquisa.

A Deus que me iluminou em todas as minhas decisões, angústias e momentos difíceis.

SUMARIO

LISTA DE FIGURAS

RESUMO

Este trabalho descreve **LAGO - Linguagem de Acesso Global ao sistema AMPLO**. AMPLO é um ambiente de projeto de sistemas digitais que consiste de uma base de dados orientada a objetos e diversas ferramentas de projeto, como editores de texto, editores gráficos, compiladores e simuladores de sistemas digitais.

LAGO é a interface de alto nível de AMPLO, sendo responsável pelo controle de acesso aos recursos do sistema. Através desta interface, os usuários de AMPLO podem ativar funções de administração, gerência de projeto, projeto propriamente dito e consulta a base de dados.

O acesso às diversas funções do sistema está vinculado à classificação do usuário. Um "administrador geral" é responsável pelo cadastro de usuários e de grupos de usuários. Grupos de usuários são criados para realizar projetos, sob a liderança de um "administrador de grupo". Este é indicado pelo administrador geral quando da criação do grupo e realiza funções de gerência de grupo e da base de dados associada ao grupo.

Usuários "projetistas" têm acesso às funções de ativação de ferramentas (editores de texto, editores gráficos, compiladores e simuladores) e às funções de administração de tarefas (uma tarefa corresponde a uma transação longa, a nível de banco de dados, por exemplo, a criação de objetos em várias sessões de edição). A cada tarefa está associada uma base de dados temporária, privativa do projetista, removida após o término da tarefa.

Portanto, a base de dados de AMPLO é dividida em três níveis: base de dados pública, bases de dados por projeto e bases de dados dos projetistas.

Permanentemente, LAGO oferece facilidades de consulta à base de dados. Estas funções de consulta estão disponíveis quando o usuário inicializa o sistema ou, posteriormente, pela seleção de uma função de consulta.

A navegação pelos objetos da base de dados pode ser feita de forma gráfica ou textual. Na forma gráfica, os objetos são apresentados através de árvores representando os diversos tipos de relacionamentos existentes. Na forma textual, LAGO apresenta listas com nomes de objetos.

PALAVRAS-CHAVE: projeto auxiliado por computador, interface com usuário, gerência de projeto, ferramentas de projeto, base de dados de projeto.

ABSTRACT

This work describes LAGO - a language for accessing the AMPLO system. AMPLO is a design environment of digital systems which is composed of an object oriented data base and of several design tools, such as text editors, graphics editors, compilers and simulators of digital systems.

LAGO is the high-level interface of AMPLO, and it is responsible for controlling the access to the system's resources. With this interface, the users of AMPLO may activate administratation and, design management functions, data base queries, and design tasks.

The access to the several system functions is in accordance to the user classification. A "general administrator" is responsible for creating users and groups of users. Groups of users create designs, under the leadership of a "group administrator", who is indicated by the general administrator when the group is created. The "group administrator" has private functions for managing the group and the data base associated to the group.

Designers activate design tools as text editors, graphic editors, compilers and simulators. Also they can use task management functions (a task corresponds to a long transaction at the data base level, like the creation of objects in several editing sessions, for example). A temporary data base which is a designer private data base is associated to each task, and is removed at the end of the task.

Thus, the data base of AMPLO is divided into three levels: public data base, group data base and designer data base.

LAGO permanently, offers facilities of data base

queries. These query functions are available when the user inicializes the system and later on, when the query function is selected.

The navigation through the data base objects can be done in a graphical or textual form. In the graphical form, the objects are presented by trees representing the several types of relationships. In the textual form, LAGO presents lists with the name of objects.

KEYWORDS: Computer aided design, user interface, design management, design tools, design data base.

1 INTRODUÇÃO

Este capítulo introduz ambientes integrados de projeto de sistemas digitais, comentando alguns aspectos pertinentes à estrutura funcional destes sistemas. A seguir, são abordados mecanismos de navegação, modelos de dados e aspectos de gerência de transações em ambientes de projeto. A partir dessa fundamentação e da breve descrição do Sistema AMPLO são colocados os objetivos do trabalho.

1.1 Ambientes Integrados de Projeto de Sistemas Digitais

O número e a complexidade de ferramentas de projeto de sistemas digitais, especialmente de circuitos VLSI, aumentaram rapidamente nos últimos anos. Paralelamente, a complexidade dos sistemas digitais projetados cresceu consideravelmente, de tal forma que começaram a ser discutidos e trabalhados aspectos como:

a) Consistência dos dados de projeto manipulados por diferentes ferramentas, através de regras de integridade, mantidas por uma base de dados completa e unificada;

b) Consistência entre diferentes representações de um mesmo sistema digital em diferentes níveis de abstração de projeto;

c) Uniformidade nas interfaces com o usuário apresentadas por diferentes ferramentas, garantindo uma resposta rápida [WEB 88], gráfica ou textual, e possuindo facilidades de manipulação de janelas, ícones, caixas de diálogo, mensagens de erros etc;

d) Mecanismos de consulta e gerência da base de dados de projeto, podendo-se visualizar e manipular os objetos armazenados;

e) Facilidades de gerência de projeto, mantendo controle de acesso [WAG 88b] ao sistema por parte de usuários, através de identificação e senha, e controle do desenvolvimento de tarefas;

f) Gerenciamento de grandes quantidades de dados durante o processo de projeto;

g) Aprendizagem dos métodos de interação com o usuário para cada ferramenta usada durante o processo de projeto;

h) Geração de ferramentas para formatos de dados específicos.

Em resposta a estas questões, inúmeros ambientes de projeto auxiliado por computador têm sido propostos (ADAM [KNA 86], ULYSSES [BUS 86], FACE [SMI 89], CWS [MIL 89], ASTA [OGI 87], FRED [WOL 86], STEM [GIR 87], entre outros). Tais sistemas, em maior ou menor grau, apresentam vantagens como: funções de gerência de projeto, consulta simples e/ou mecanismos de navegação na base de dados, interface homogênea entre as ferramentas, funções de acesso global ao ambiente em qualquer nível, troca automática de informações entre ferramentas, representações de dados equivalentes entre ferramentas, integração entre ferramentas de aplicação e controle de versões.

1.1.1 CWS

CWS é um ambiente integrado de CAD, consistindo de um conjunto de sub-ambientes de aplicação e um único sub-ambiente de gerência de projeto. A integração dos sub-ambientes no CWS é feita através de uma abordagem orientada a objetos, que permite extensão e modificação de ferramentas.

As principais características da abordagem orientada a objetos do **CWS** são: **abstração de dados** (tipos de

objetos são definidos por suas estruturas de dados e um conjunto de métodos que agem sobre objetos deste tipo), **polimorfismo** (métodos com o mesmo nome e significado podem ser executados sobre objetos de diferentes tipos), **herança** (um tipo de objeto pode ser definido com base na especificação de um outro tipo de objeto previamente definido) e **representação** (instâncias de objetos e métodos podem ser associados a uma instância gráfica).

O ambiente de projeto **CWS** consiste de um sistema de manipulação de dados (**DHS - "Data Handling System"**), um sistema de interface com o usuário (**UIS - "User Interface System"**) e um sistema de integração orientado a objetos (**OIS - "Object Oriented Integration System"**). As ferramentas de aplicação incluem um editor de esquemáticos, um gerenciador de "net-list" e simuladores. CWS é escrito em linguagem "C", num ambiente "UNIX", usando o sistema X-Windows.

A principal tarefa do **DHS** dentro do **CWS** é manter os tipos de objetos que constituem o ambiente, suportando instanciamento e manipulação dos objetos, de tal forma que todos os aspectos de projeto eletrônico podem ser representados, permitindo também navegação na base de dados. O **UIS** fornece ao projetista uma clara visão do projeto em que ele está trabalhando, permitindo navegação através dos objetos, suportando e controlando suas ações sobre o projeto.

A principal tarefa do **OIS** dentro do **CWS** é manipular os objetos e métodos que constituem o ambiente de projeto. Possui mecanismos que permitem integração, extensão ou modificação das ferramentas de projeto. Mantém os conceitos de objeto e gerencia a lista de usuários correntes.

As principais vantagens da abordagem orientada a

objetos do ambiente **CWB** são: promove projeto efetivo da integração pela separação forçada entre projeto e implementação da integração, suporta uma metodologia de integração que é completamente consistente com a metodologia de interação com o usuário, separa a semântica dos sub-ambientes integrados dos mecanismos que os manipulam e não faz distinção entre os sub-ambientes de gerência de projeto e qualquer outro sub-ambiente.

1.1.2 FACE

O ambiente de compilação de arquitetura flexível **FACE** ("Flexible Architecture Compilation Environment") tem um núcleo desenvolvido usando metodologia orientada a objetos que pode ser rapidamente adaptado a aplicações específicas. O núcleo provê um modelo formal para representar estruturas de dados baseadas em grafos e hipergrafos, algoritmos que operam este modelo, ferramentas de desenvolvimento de aplicativos de alto nível, como editores interativos, linguagem de consulta, recuperação e armazenamento de dados persistentes, gerenciamento de configurações de projeto e uma interface com o usuário para ativação de ferramentas de aplicação. Estas ferramentas podem ser chamadas interativamente pelo usuário ou pelo controle automático de funções da ferramenta de alto nível.

Uma nova ferramenta de aplicação é implementada no ambiente **FACE** pelo mapeamento das classes de objetos abstratos de dados manipulados pela ferramenta nas classes de objetos fornecidas pelo núcleo.

Uma exigência fundamental em CAD é a habilidade para criar estruturas internas com dados de projeto a partir de informações externas. O ambiente **FACE** provê um conjunto de interfaces funcionais para simplificar o desenvolvimento de analisadores sintáticos de linguagens como SPICE, HILO, EDIF etc.

1.1.3 ADAM

ADAM ("Advanced Design Automation") é um sistema de automação de projeto avançado. É, basicamente, um conjunto de rotinas / programas para gerência de projeto. A base de conhecimentos é mantida em RTL ("Register Transfer Level"), ou seja, a nível de transferência entre registradores.

O sistema **ADAM** é usado para integrar inúmeros programas de automação de projeto em um único ambiente, como, por exemplo, geração automática de "lay-out", onde um sistema especialista é usado para testar regras de projeto de circuitos. Todos os dados de projeto, procedimentos e regras são tratados como objetos; a comunicação entre objetos é feita por meio de mensagens.

Os objetivos de **ADAM** são:

a) produzir implementações cuja correção possa ser testada;

b) permitir graus variáveis de interação com o usuário;

c) permitir projeto incremental, isto é, projeto iniciando a partir de outro, parcialmente completo;

d) garantir projeto dentro de diversas espécies de restrições.

ADAM usa tanto técnicas convencionais como técnicas baseadas em conhecimento para atingir estes objetivos. Conforme já foi mencionado, **ADAM** é usado para gerenciar atividades de projeto. Primeiramente um plano de projeto é construído, o qual apresenta detalhes das atividades e facilidades que serão usadas; em uma segunda fase, o plano é executado. Se a execução do plano for bem sucedida, o projeto está correto e mantém as restrições. Caso contrário, **ADAM** tenta construir um novo plano. Portanto, este sistema controla tanto o processo de execução como o processo de planejamento do projeto.

ADAM define um conjunto fixo de visões de dados para representar a informação de projeto. Estas visões servem como uma base de representação para ferramentas de aplicação, mas não possui mecanismos para estender a informação necessária em feramentas de aplicação específicas.

1.1.4 ULYSSES

ULYSSES é um ambiente destinado à solução de problemas associados com integração de ferramentas de CAD para VLSI. Especificamente, **ULYSSES** permite a integração de ferramentas de CAD para um sistema de automação de projeto, a especificação de uma metodologia de projeto e a representação do espaço de projeto. Por espaço de projeto, em ULYSSES, entende-se todos os projetos possíveis para um dado problema, ou seja, todos os projetos que atingem, de alguma forma o objetivo desejado. O ambiente emprega técnicas de inteligência artificial, funcionando como um sistema especialista interativo, e interpreta descrições de tarefas de projeto escritas em uma linguagem do tipo roteiro.

ULYSSES faz distinção entre uma ferramenta de CAD que executa uma função simples tal como simulação de circuito, e um sistema de automação de projeto que é capaz de gerar um "lay-out" de circuito integrado, dada uma descrição comportamental do mesmo. Tal sistema de automação de projeto pode ser obtido a partir do desenvolvimento de um ambiente de projeto que pode integrar ferramentas de CAD individuais.

ULYSSES usa uma linguagem de "scripts" para codificar tarefas de projeto bem sucedidas. **ULYSSES** permite ao projetista fazer trocas arbitrárias no projeto e restaura a consistência de projeto pela propagação automática dos efeitos das trocas em todos os arquivos relacionados. Finalmente, **ULYSSES** emprega um compilador de roteiros para

gerar automaticamente regras de produção de controle de ferramentas.

1.1.5 ASTA

ASTA é um sistema de gerenciamento de projeto de circuitos VLSI que automatiza a análise de desempenho de ferramentas de CAD, suporta o controle de versões de bibliotecas e ferramentas e o controle do processo de projeto, permite estimativa de custo de projeto e fornece uma coleção de informações estatísticas como, por exemplo, tamanho do circuito que está sendo projetado. A automação do gerenciamento de projeto é acompanhada pela análise estatística e pelas informações de controle que são geradas por todas as ferramentas e pelo sistema operacional.

Como já visto nos ambientes mencionados, todos eles, em geral, possuem aspectos de gerência de projeto, interface uniforme com o usuário, consulta (navegação) à base de dados e integração das ferramentas de aplicação. Poucos, entretanto, são ambientes abertos com mecanismos que suportam a integração de outras ferramentas.

1.2 Mecanismos de Navegação ("browsers") em Bancos de Dados de Projeto

"Browsers" em bancos de dados convencionais são ferramentas que permitem consultar e alterar os valores de campos selecionados. Inicialmente, percorriam sequencialmente (ou segundo algum índice) a base de dados mostrando os conteúdos vigentes e permitindo movimentação em torno do registro corrente. Com a introdução de interfaces seguindo o paradigma da manipulação direta, o usuário pode realizar o exame de objetos na vizinhança semântica do objeto de interesse. Por exemplo, é possível focalizar um objeto que é propriedade do objeto corrente, focalizar sua classe etc [LAE 89].

O conceito de "browser", entretanto, como é salientado em [CZE 90] não é o de uma linguagem de consulta completa e sim de um guia para exibir ítens requisitados. Um "browser" simples provê um símbolo ou ícone para cada tipo de objeto que pode aparecer na base de dados e várias ações que permitem, tipicamente, mudar o foco de atenção do usuário de um objeto para outro que apresenta alguma relação semântica com ele [LAE 89]. Exemplo típico é encontrado no sistema **WHITEBOARD** [DON 86], onde representações dos quadros brancos ("Whiteboards") comuns nos escritórios são armazenadas numa base de dados.

Cada base de dados no **WHITEBOARD** contém arquivos textos, notas, programas, ferramentas, figuras etc, como uma coleção de ítens não necessariamente relacionados. A organização espacial dos ítens na janela também é armazenada.

A partir do armazenamento da disposição dos ítens de uma janela de "whiteboard" numa base de dados, o "browser" permite que uma visão seja reconstituída. "Browser" também é encarregado da abertura de um ícone, expansão

("zoom") de um "whiteboard" e reposição de janelas. Por abertura de ícone se entende a ativação da janela ou programa representado por ele.

A introdução de "browsers" em ambientes de projeto permite explorar interativamente as estruturas complexas (ex: grafos) que representam os objetos em aplicações de CAD [GED 88]. Assim sendo, este tipo de ferramenta aumenta a facilidade de reutilização de projetos pela facilidade de localização dos mesmos e, além disso, melhora a comunicação e compartilhamento entre diferentes membros da equipe de projeto. As várias relações existentes entre objetos num ambiente complexo como o de CAD podem ser utilizadas como "caminhos" do "browser".

Exemplos típicos de tais relações são "é_componente_de", "é_composto_de", "é_ancestral_de", "é_descendente_de" e "é_equivalente_a". Claramente, a maioria delas origina árvores ou grafos. E objetivo do "browser", portanto, oferecer meios de navegação através de árvores ou grafos; tais meios devem tornar fácil e rápida a tarefa de especificar ou alcançar o próximo objeto no contexto.

Um modo de especificar o objeto de interesse é pelo seu nome. Se várias janelas estiverem sendo usadas, todas devem ter seu conteúdo atualizado em função do novo objeto. A exploração das relações entre objetos como "caminhos" do "browser" leva a outro tipo de especificação, que é o de apontamento com "mouse": qualquer objeto que está sendo exibido, sendo indicado com o "mouse", passa a ser o próximo objeto corrente (de interesse). As estruturas visualizadas em uma ou mais janelas são atualizadas: grafos são redesenhados e caixas de diálogo são reexibidas.

O sistema **WHITEBOARD**, por exemplo, foi utilizado para construir uma base de dados de documentação do sistema **CEDAR** [TEI 84]. Caixas de texto num determinado nível de

detalhe são exibidas em conjunto com ícones que permitem ativar outras "páginas" do texto.

Em aplicações de projeto, mesmo focalizando um único objeto, as estruturas exibidas podem ser bastante complexas, e o número de objetos exibidos pode ser bastante grande. Os mecanismos tradicionais de "zooming" e "panning" utilizados em editores gráficos não se aplicam efetivamente ao problema de redução de objetos em cena. E necessário um mecanismo de seleção de elementos a serem exibidos, uma espécie de "poda" das árvores e grafos. Esse mecanismo equivale à escolha de um sub-grafo/sub-árvore e corresponde à seleção de um conjunto suficientemente pequeno de objetos relacionados com aquele em questão.

[GED 88] apresenta duas formas de especificar o relacionamento que determina a "poda": "poda retangular" e "poda ampulheta". O primeiro caso, com retângulo especificado em termos de **altura**, **largura** e **posição**, é usado para construir um sub-grafo que tenha o objeto de interesse em **posição**, e outros objetos ancestrais e descendentes dependendo da altura e largura do retângulo. Estes valores de posição, altura e largura correspondem a níveis de nodos no grafo. Assim, o usuário pode selecionar níveis intermediários do grafo, centrando o retângulo no nível do objeto de interesse. O segundo caso é similar à "poda retangular" mas limita os objetos que aparecem no sub-grafo a ancestrais e descendentes diretamente ligados ao objeto de interesse. Por isso, o grafo toma a forma de uma ampulheta, com o objeto de interesse no centro.

Consultas a uma base de dados de projeto são freqüentemente mais complexas de especificar e implementar do que num banco de dados convencional, uma vez que a estrutura subjacente dos dados envolvidos em aplicações de projeto é muito mais rica e complexa. Por isso linguagens textuais de consulta são freqüentemente usadas. O envolvi-

mento de atributos dos objetos como fator de seleção, além das relações fundamentais mencionadas anteriormente, não foi suficientemente explorado graficamente por [GED 88]. Estes autores apenas mencionam que processos de "poda" podem agilizar a resposta a questões envolvendo atributos diversos.

A diversidade de possibilidades de consulta originou tipos diferentes de "browsers" no sistema **NEPTUNE** [DEL 86], destinado à construção e exibição de documentos hierárquicos (hipertextos). Existem três tipos básicos de "browser" em **NEPTUNE**: ("graph browser", "document browser" e "node browser").

O "**Graph Browser**" mostra uma visão gráfica de um hiperdocumento ou uma porção de um hiperdocumento. Um hiperdocumento é exibido como um grafo. Cada nodo é representado por um ícone que consiste de um nome dentro de um retângulo. A janela onde o grafo é exibido é dividida em quatro áreas: a área superior contém a visão do grafo, a área esquerda é uma área de "scroll" para "zoom" e "pan", e as duas subjanelas horizontais inferiores contêm editores de texto usados para definir os predicados de visibilidade de nodos e elos.

O "**Document Browser**" é usado para simplificar a manipulação de hiperdocumentos estruturados hierarquicamente. Possui cinco subjanelas: as quatro superiores contêm as listas dos nomes dos nodos; a inferior, maior, é o "Node browser" que pode ser usado para exibir o conteúdo de um dos nodos listados nas janelas superiores. Cabe comentar que **NEPTUNE** é implementado em **SMALLTALK-80** [GOL 84], tendo portanto todas as características da interface deste sistema.

O "**Node Browser**" permite que o conteúdo de um nodo individual seja editado, suportando navegação via elos e a

criação de nodos elos.

Uma abordagem diferente, tanto para consulta como para atualização de bases de dados, é adotada por [CZE 90]. Neste trabalho, os autores editam diagramas entidade-relacionamento estendidos para especificar consultas: as operações sobre o diagrama são utilizadas sobre o conjunto de dados para a realização da consulta. O diagrama que representa o esquema da base de dados é exibido; operações de remoção, restrição, junção são utilizadas sobre o diagrama e colecionadas. Ao se obter o diagrama que representa a resposta da consulta, as operações são realizadas sobre a base de dados e o resultado é exibido. De forma análoga, atualizações podem também ser especificadas.

Tanto [GED 88] como [CZE 90] ainda comentam aspectos de interfaces gráficas para consulta a bases de dados gerais. [LAE 89] apresenta uma interface específica para desenvolvimento de aplicações sobre um sistema de banco de dados orientado a objetos.

1.3 Modelos de Dados e Mecanismos de Gerência de Transações em Ambientes de Projeto

Segundo [HAR 89], um modelo de dados que suporte a definição e manipulação de objetos complexos deve apresentar as seguintes características:

- permitir a representação de todos os relacionamentos essenciais presentes na realidade a ser modelada;

- permitir a representação de objetos recursivos. A estruturação interna de objetos complexos leva, naturalmente, à necessidade de construções e, consequentemente, manipulações de estruturas de dados complexas;

- permitir a definição dinâmica de objetos. Aplicações não convencionais, como as de projeto, necessitam definir e manipular diferentes visões (níveis de abstração) de um mesmo objeto. As definições estáticas (definidas no esquema da base de dados em tempo de modelagem da aplicação) não oferecem tal flexibilidade. Definições dinâmicas de objetos podem ser conseguidas por meio da utilização de uma linguagem de consultas que permita a construção de tipos de dados derivados a partir de outros já existentes;

- permitir a simetria de relacionamentos. Os relacionamentos entre objetos devem ser representados direta e inversamente a fim de permitir o processamento de conjuntos de objetos de forma multidirecional;

- oferecer suporte ao tratamento de versões. Casos em que aspectos temporais mereçam registro ocorrem com frequência em aplicações de projeto onde é muitas vezes necessário manter um histórico dos passos percorridos, inclusive por questões de segurança.

E desejável ainda, que o modelo de dados também

apresente uma orientação comportamental a objetos. Por exemplo, através da definição de tipos abstratos de dados, pode-se embutir, no modelo, a verificação de restrições de integridade complexas que garantam restrições semânticas específicas da aplicação sendo modelada.

Projetos de engenharia são, em geral, muito complexos para serem realizados integralmente por uma única pessoa [DIT 86]. É fato comum distribuir as tarefas de projeto entre os vários projetistas da equipe. A metodologia de projeto envolve os seguintes detalhes [DIT 86]:

- partes de um projeto que apresentem algum grau de completeza e consistência podem ser colocadas à disposição de outros projetistas;

- projetos ou partes de projetos que se encontrem parcialmente completos mas necessitem do trabalho de outros especialistas podem ser cedidos a estes para que os completem;

- projetos que não se encontrem em condições de serem compartilhados devem permanecer fora do alcance dos demais usuários.

Nos casos de transferências de objetos, o projetista proprietário pode especificar se está cedendo o objeto ou apenas emprestando. Em caso de empréstimo o proprietário deve recebê-lo de volta posteriormente.

Segundo [HAR 89], uma arquitetura de sistema de gerência de banco de dados para apoio a atividades de projeto deve prover, entre outros, os seguintes requisitos:

- **permitir manipulação eficiente de objetos**. As aplicações, em execução nas estações de trabalho ("workstations"), devem minimizar os acessos a dados armazenados nos servidores;

- **suprimento de dados**. Os custos de comunicação e tranferência de dados entre estações de trabalho e servidores têm um grande impacto sobre o desempenho do sistema. Assim sendo, deve-se procurar minimizar o número de mensagens trocadas entre estações de trabalho e servidores e procurar realizar as transferências de dados com o maior volume possível. Deve-se, ainda, manter nas estações de trabalho, bases de dados privativas que armazenaram os dados utilizados pelas aplicações locais em execução. O uso de bases de dados privativas conferem às estações de trabalho maior autonomia e tolerância a falhas;

- **isolamento contra falhas**. É desejável que em caso de falha no sistema, a quantidade de trabalho a ser desfeita e/ou refeita seja mínima. Também espera-se que uma falha ocorrida em uma estação de trabalho possa ser recuperada pela própria estação de trabalho, sem gerar sobrecarga de trabalho para os servidores e vice-versa. Por esta razão, grandes esforços são dispendidos no sentido de esconder, mutuamente, as falhas ocorridas nas estações de trabalho e nos servidores.

Este último ítem diz respeito ao mecanismo de transações adotado. Uma transação [DAT 77] é uma unidade de trabalho (sobre um banco de dados) com a propriedade de que o banco de dados (a) está num estado consistente antes e depois dela mas (b) possivelmente não está em tal estado durante ela. Uma transação corresponde a um conjunto atômico de operações que modificam o estado do banco de dados. Uma transação convencional inclui uma sequência de interações que o usuário mantém com a base de dados. Esta sequência deve ser executada atomicamente e deve refletir a idéia de que as ações do usuário são isoladas de quaisquer outras atividades concorrentes. A execução de uma transação é dita correta se o estado do banco de dados atingido após sua execução satisfaz o conjunto de restrições de integridade associadas a ele [BAN 86].

Transações de projeto se caracterizam por serem de longa duração (horas, dias ou semanas), e necessitam após um determinado tempo, avaliação na integridade e consistência dos dados [HAR 83]. São compostas por transações tradicionais e/ou outras transações longas.

As bases de dados, em um ambiente de projeto, são, em geral, organizadas de forma hierárquica. Tais hierarquias costumam envolver três níveis: público, de projeto e privativo. A base de dados pública contém dados acessíveis a todos os projetistas; bases de dados de projeto têm seus dados, via de regra, acessados pelos participantes dos respectivos projetos; bases de dados privativas contém dados privados de cada usuário.

Genericamente, as transações tradicionais realizam leituras e atualizações sobre bases de dados privativas. Transações de usuários lêem e atualizam dados das bases privativas, podendo, eventualmente, requisitar dados presentes nas bases de dados de projeto e liberar objetos consistentes para bases de dados de projeto. Os objetos requisitados devem ser devolvidos ao final da transação. Transações de projeto, por sua vez, são conjuntos de transações de usuário, podendo, também, requisitar objetos à base de dados pública. Os objetos requisitados devem ser devolvidos antes do encerramento da transação de projeto.

Uma transação longa pode encontrar-se em um dos três estados seguintes [HAR 88]:

- **inexistente**: Estado que precede o início ou sucede o final da transação longa;

- **ativo**: A transação assume este estado quando de sua ativação, ou quando suspensa, seja reativada;

- **suspenso**: Estado atingido quando, em um estado ativo, a transação é temporariamente suspensa. Enquanto suspensa, a transação preserva todos os bloqueios detidos sobre os objetos que foram copiados de um nível superior de hierarquia de bases de dados.

A seguir, na descrição do AMPLO, serão vistos como estes aspectos são tratados neste sistema em particular.

1.4 Descrição Geral do Sistema AMPLO

O sistema AMPLO (AMbiente integrado para Projeto LOgico de sistemas digitais) [WAG 88, FRE 88] é um ambiente integrado para o desenvolvimento de sistemas digitais, em desenvolvimento na UFRGS. Este ambiente possui ferramentas que permitem a descrição e validação de sistemas digitais em três níveis de projeto: sistema, transferência entre registradores (RT) e portas lógicas. Para cada um destes níveis existe uma linguagem de descrição de hardware.

Neste ambiente, sistemas digitais são descritos modular e hierarquicamente como **redes de agências**. Cada agência na rede pode ser descrita de duas formas:

- como uma agência **Composta**, constituída por outra rede de agências, através da linguagem **REDES** [WAG 87a], nada constando a respeito da implementação das sub-agências;

- como uma agência **Primitiva**, através de uma das três linguagens de descrição de hardware: LAÇO [SIL 88], uma linguagem baseada em redes de Petri, para descrições a nível de sistema, KAPA [WAG 87c], uma linguagem para descrições estruturais no nível de transferência entre registradores e NILO [WAG 87b], uma linguagem para descrições estruturais a nível de portas elementares.

Todas as quatro linguagens suportadas pelo ambiente possuem forma gráfica e textual que são completamente equivalentes entre si [WAG 86a]. Isto permite que a descrição de uma agência possa ser feita de forma textual, via editor de textos convencional e compilador específico, ou de forma gráfica, via um editor gráfico especializado para a linguagem. Tanto o compilador como o editor gráfico geram a mesma estrutura de dados interna, que é independente da forma de entrada, exceto pelas informações geométricas manipuladas pelo editor. Esta representação interna é armazenada em uma

base de dados unificada, de onde pode ser recuperada para construção de modelos simuláveis [WAG 86b].

AMPLO permite ao projetista de hardware a criação de diversas alternativas e versões de projeto de um mesmo sistema (agência).

Alternativas de uma agência diferem entre si por apresentarem diferentes definições para a interface. Alterações na interface que criam uma nova alternativa de projeto para uma agência são: inclusão ou remoção de sinais; mudança no nome e na largura (tamanho) em "bits" de um sinal; mudança do tipo de dados de um sinal e mudança de atributos geométricos da interface. Uma mesma alternativa de uma agência pode estar definida em diversos níveis de descrição. Estas definições devem ter mesmo número de sinais e mesmos atributos geométricos para a interface. Cada um dos sinais da interface deve ter, em todas as definições, mesmo nome e mesma largura em "bits". Os tipos de dados associados a um sinal nas diversas definições de uma mesma alternativa devem ser compatíveis entre si.

A cada alternativa de uma agência podem estar associadas diferentes **versões** de projeto. Versões de uma mesma alternativa têm portanto em comum uma mesma definição para a interface (ou definições compatíveis, se feitas em níveis diferentes). Cada alternativa de uma agência pode ter qualquer número de versões, tanto compostas como primitivas. Uma versão composta é uma descrição em REDES e uma descrição primitiva é uma descrição em qualquer uma das linguagens LAÇO, KAPA ou NILO.

Alternativas e versões de projeto são identificadas por números inteiros. Uma alternativa de uma agência é, portanto, designada por: "⟨nome_da_agência⟩.⟨número_da_alternativa⟩", enquanto uma versão é identificada por: "⟨nome_da_agência⟩.⟨número_da_alternativa⟩.⟨número_da_ver-

são". Cada tipo de agência criado na base de dados corresponde a uma alternativa de projeto de uma agência. Qualquer versão desta alternativa pode ser utilizada no lugar de uma ocorrência deste tipo, já que todas as versões de uma mesma alternativa têm a mesma interface, o mesmo não ocorrendo para versões de alternativas diferentes. Neste caso, a versão composta REDES, descrita em termos de ocorrências de versões, corresponde a uma configuração estática da agência. Numa descrição REDES podem ser usadas, por outro lado, ocorrências de alternativas. Por ocasião da simulação são escolhidas as versões a serem incluídas na descrição para efeito de simulação, correspondendo a uma configuração dinâmica.

Projetos são validados por simulação. AMPLO possui uma família de simuladores: três simuladores para modelos consistindo de agências que estão todas descritas numa mesma linguagem primitiva e um simulador multi-nível [WAG 86b] para modelos consistindo de agências descritas em diversos níveis de projeto.

Todos estes simuladores são construídos baseados num princípio mestre-escravo [WAG 86b, WAG 87d]. O algoritmo mestre é responsável pelas interações entre as agências na rede, enquanto o algoritmo escravo é responsável pelas atividades no interior das agências. Existe um algoritmo escravo especializado para cada uma das linguagens primitivas de AMPLO.

Todos os dados a respeito dos sistemas digitais especificados em AMPLO são armazenados numa base de dados acessada via uma interface orientada a objetos [GOL 88]. Essa interface corresponde a um conjunto de primitivas de acesso que garantem a consistência dos dados armazenados nos diversos arquivos utilizados para implementar fisicamente a base de dados. Os objetos tratados por esta interface são tipos de agências (onde cada tipo é uma alternati-

va de projeto de uma agência), versões para cada alternativa, modelos de simulação, etc. Esses objetos possuem atributos tais como interface (sinais, tipos de dados, etc) e geometria. Entre os objetos existem múltiplas relações. Por exemplo, uma agência composta contém ocorrências de outros tipos de agência; um modelo de simulação é uma rede de versões de agências; etc.

No sistema AMPLO, a base de dados é dividida em: pública, de projeto e privativa [NET 90]. A base de dados pública é permanente e permite acesso aos dados a todos os usuários cadastrados, enquanto que a de projeto é temporária e só permite acesso aos usuários do projeto. Na base de dados privativa, que também é temporária, o acesso é somente do projetista. Através da interface de alto nível do sistema, **LAGO** (**L**inguagem de **A**cesso **G**lobal ao sistema AMPL**O**), os usuários podem consultar as bases de dados e realizar tarefas que envolvem empréstimo, transferências, devolução e liberação de objetos. Todas estas operações são citadas em detalhes no capítulo 3.

O modelo de processamento do sistema AMPLO [NET 90] provê três tipos de transações:

- **transação de projeto**: envolve o trabalho de um grupo de projetistas no desenvolvimento de um projeto comum; caracteriza-se como uma transação longa;

- **transação de usuário**: representa o trabalho de cada projetista participante do projeto; caracteriza-se como uma transação longa;

- **transações curtas**: execução de uma das primitivas de gerência de projeto, consultas rápidas a níveis superiores da hierarquia de bases de dados.

Transações de projeto e de usuário estão associa-

das a bases de dados de projeto e privativas, respectivamente, que são extintas, quando estas transações encerram.

Durante uma sessão de trabalho, o usuário pode requisitar objetos às bases de dados, devendo o sistema gerenciador de banco de dados prover transparência de localização, ou seja, o usuário não precisa especificar em qual das bases de dados da hierarquia o objeto requisitado se encontra. Assim sendo, caso o objeto desejado não esteja presente na base de dados privativa, o gerenciador que está executando a primitiva de solicitação se encarrega de refazer a solicitação nos níveis superiores da hierarquia de bases de dados.

Transações de usuários podem ser temporariamente suspensas e posteriormente retomadas. A suspensão de uma transação de usuário implica na definição de um ponto de salvamento gerado automaticamente pela primitiva de suspensão. O usuário também pode definir pontos de salvamentos explícitos, que podem ser internos ou externos. Pontos de salvamento externos são efetivados pela execução da primitiva de salvamento. Esta deve ser acionada num instante em que nenhuma transação curta esteja em execução no escopo da transação de usuário. Pontos de salvamento internos devem ser requeridos pelos aplicativos.

A metodologia de projeto preconizada em AMPLO faz uso de conceitos como usuário, senha, grupo, contexto, projeto, tarefa e níveis de banco de dados. Um usuário especial de AMPLO, o administrador geral, faz uso de suas funções de gerência para cadastrar os demais **usuários** e criar **grupos de usuários**, designando **administradores**. Cada usuário pode estar vinculado a um ou mais grupos como projetista. Os grupos são responsáveis pelo desenvolvimento de **projetos**. Cada projetista, na consecução do(s) projeto(s) em que está engajado, realiza **tarefas**, que correspondem a atividades de especificação, construção e validação

de objetos.

Para a realização das tarefas, os projetistas podem utilizar objetos previamente existentes no banco de dados (público) do AMPLO. A definição dos objetos aos quais os projetistas de determinado grupo têm acesso é feita através do conceito de **contexto**, que implementa visões sobre o banco de dados público. Contextos são definidos como regras ou por enumeração de objetos e são associados a grupos e a projetos, permitindo, então, sua utilização por parte dos projetistas. Cada projetista, por sua vez, possui um banco de dados privativo, vinculado à tarefa que está realizando.

1.5 Motivação, Objetivos e Apresentação

AMPLO suporta o projeto integrado de circuitos com base em uma interface homogênea entre o usuário e todas as ferramentas. Essa homogeneidade tem sido buscada em dois níveis:

- ferramentas de uma mesma classe apresentam a mesma interface com o usuário;

- todas as ferramentas são acessadas a partir de uma interface gráfica de alto nível.

O primeiro nível de homogeneidade é atingido pelo projeto e implementação das ferramentas utilizando os mesmos pacotes gráficos e de gerência de janelas. Assim, todos os editores compartilham da mesma organização de tela, dos mesmos mecanismos interativos básicos e, inclusive, de certas operações sobre objetos; os simuladores são ativados a partir de um ambiente unificado de simulação e os compiladores apresentam, também, mesmo estilo de interação com o usuário. Para atingir tal homogeneidade, AMPLO possui um conjunto de rotinas gráficas [OLA 88], divididas em duas partes: um subconjunto é dependente de dispositivo (**IE** - **I**nterface gráfica de **E**ntrada [OLA 87b] e **IS** - **I**nterface gráfica de **S**aída [OLA 87a, OLA 87c]) e outro, independente de dispositivo (**PG** - **P**acote **G**ráfico do sistema AMPLO [PIN 88]), é orientado às necessidades dos editores de AMPLO. Com base neste pacote independente de dispositivo opera um conjunto de funções para a especificação das características básicas dos diálogos que as ferramentas do sistema estabelecem com o usuário (**PIU** - **P**acote de controle da **I**nterface com o **U**suário [OLA 89]).

Para gerar a interface de uma ferramenta no sistema AMPLO, faz-se uso da linguagem **LINUS** (**Linguagem de especificação de INterface com o USuário** [OLA 89]), onde são

descritos os atributos das áreas de comunicação gerenciadas pelo PIU.

Obter o segundo nível de homogeneidade, ou seja, especificar e desenvolver a interface de alto nível de AMPLO, é o objetivo do presente trabalho. Para tanto é necessário definir uma linguagem gráfica de comandos global e projetar e implementar seu interpretador, integrado às diversas ferramentas existentes no ambiente AMPLO.

LAGO, a Linguagem de Acesso Global ao amplO, permite a ativação de editor de texto, compiladores e editores gráficos específicos das linguagens de descrição de hardware, a passagem para o ambiente de simulação e diversas tarefas de gerência de projeto. Consultas à base de dados podem ser realizadas sob forma de navegação na estrutura de dados hierárquica das agências.

Os demais capítulos deste trabalho estão organizados como segue. No capítulo 2 são abordados os aspectos pertinentes às interfaces para ambientes integrados de projeto. O capítulo 3 descreve LAGO, a interface gráfica de alto nível específica de AMPLO, apresentando as funções de gerência de projeto e mecanismo de navegação na base de dados. No capítulo 4, a implementação feita é descrita, abordando-se, também, as funções que foram prototipadas sobre arquivos convencionais, em virtude do estágio atual de desenvolvimento do acesso ao banco de dados. Finalmente, no capítulo 5, são apresentadas a avaliação do trabalho e conclusões pertinentes acerca do mesmo. Os apêndices contém detalhes da implementação e organização do software. O apêndice A corresponde à especificação de LAGO na linguagem de entrada para o Gerador de Interfaces orientadas a Menus (GIM [BAG 89b]); o apêndice B comtempla aspectos da compilação do software; o apêndice C reúne as descrições das estruturas de dados mais importantes de LAGO e, finalmente, o apêndice D descreve todas as funções de LAGO, a nível de

entradas, saídas e funcionamento.

2 INTERFACES PARA AMBIENTES INTEGRADOS

2.1 Interfaces com o Usuário

A implementação de interfaces amigáveis e de alto nível tem sido facilitada pela melhora nos sistemas de computação e por máquinas mais eficientes.

Hoje, através dos inúmeros periféricos de entrada e saída disponíveis, torna-se imperativa a implementação de interfaces interativas, em aplicações de projeto auxiliado por computador [WEB 88].

Deve-se tomar cuidado com as facilidades implementadas, pois elas podem acabar dificultando o uso da interface e o aprendizado. O sistema deve encaminhar o usuário para que ele visualize o trabalho que está sendo executado, através de mensagens de erros, ícones, caixas de diálogo, janelas sobrepostas, menus e outras facilidades.

Deve-se esperar uma resposta rápida [WEB 88] gráfica ou textual, ou mensagens que indiquem a causa da demora, para que a interface seja eficiente e confiável, não deixando de ser amigável.

No caso de ambientes integrados, é necessária uma homogeneidade entre as ferramentas acessadas pela interface, para melhor manuseio pelo usuário.

2.2 Avaliação de uma Interface

Podemos adotar os seguintes parâmetros para avaliar uma interface [SHN 83]:

a) Tempo de aprendizado do usuário na utilização dos comandos da interface;

b) Velocidade do usuário em executar uma tarefa;

c) Taxa de erros, ou seja, quantos e quais tipos de erros o usuário comete;

d) Satisfação encontrada pelo usuário na manipulação da interface;

e) Tempo de retenção, ou seja, quanto tempo o usuário consegue manipular a interface sem esquecer seu uso.

Dependendo do tipo de sistema, alguns destes parâmetros devem ser mais importantes. Os projetistas devem escolher as melhores opções. Deve-se confeccionar manuais e documentos de especificação bem claros e organizados. Pode-se ter também, uma função "help", ou seja, uma função que "ajuda" o usuário a manipular a interface, que é ativada e desativada facilmente.

Para sistemas de projeto auxiliado por computador, a simplicidade de aprendizado e a satisfação do usuário são elementos prioritários para que o usuário manipule a interface.

2.3 Tipos de Usuários

Para especificar e implementar uma interface, deve-se levar em consideração o tipo de usuário. Um sistema interativo tem por objetivo facilitar a manipulação por parte dos usuários.

As diferenças entre os usuários são importantes, e eles podem ser classificados, por exemplo, como iniciantes, experientes e ocasionais. Para cada tipo de usuário tem-se diferentes tipos de objetivos, desde informações simples a comentários mais detalhados [FOL 82].

2.4 Modos de Interação [SHN 87]

Não há um modo de interação que sirva a todos os usuários; deve-se levar em consideração o tipo de usuário que se quer atingir. Interfaces estruturadas em camadas facilitam ao usuário inexperiente: na primeira camada funções simples e, nas próximas, funções cada vez mais complexas.

Quando o sistema for usado com frequência, é aconselhável o uso de uma linguagem de comandos. O usuário pode executar uma tarefa mais rapidamente se tiver bom conhecimento da linguagem. Um problema é o tempo de aprendizado da linguagem. Este tipo de sistema é restrito a usuários experientes.

Para usuários ocasionais o uso de menus é bem sucedido. Menu é uma lista de opções, da qual o usuário pode selecionar, em geral, uma. Pode-se ter menus interligados, estruturados em árvore na qual o primeiro nível de menu é a raiz.

Menus podem aparecer em janelas sobrepostas, onde as opções são selecionadas por barras reversas, caracteres de identificação, cursor acionado a partir de periféricos como "mouse" ou "tablet".

Quando o periférico utilizado é o teclado, para facilitar a manipulação do sistema pelo usuário deve-se cuidar a organização e padronizar a programação das teclas. Em especial, a programação das teclas deve ser compatível com a de outros sistemas utilizados frequentemente pelo usuário.

2.5 Aprendizado do Uso da Interface

O aprendizado deve ser fácil, não necessitando uso de manuais ou da função "help". Preferencialmente, o usuá-

rio deve aprender automaticamente com o uso. O sistema deve informar ao usuário seus passos, erros e demoras. Para usuários inexperientes deve-se ter "help" ativo (ou seja, uma janela de "ajuda", que contém informações básicas sobre o sistema. Está função pode ser desativada após o aprendizado por parte do usuário), teclas de retorno e mensagens de erro.

Não se deve colocar muitas informações ao mesmo tempo na tela. O usuário deve poder, ao solicitá-las, sentir necessidade de mais informações. Todas as ações do usuário devem ter uma resposta clara e serem reversíveis.

3. LAGO - a Interface de Alto Nível do Sistema AMPLO

3.1 Introdução Geral

3.1.1 Características Gerais

LAGO é uma linguagem de comandos orientada por cardápios. A partir de opções exibidas na tela, podem ser acionadas funções de gerência de usuários e de grupos de usuários, funções específicas de projeto (edição, compilação, simulação) e de controle de projetos e tarefas. São oferecidas, também, várias formas de consulta ao banco de dados.

Cabe ressaltar a distinção entre projeto e tarefa. Um projeto está, em geral, associado ao desenvolvimento de um sistema digital (complexo ou não). Projetos são subdivididos em tarefas de longa duração ou não, que correspondem a edições, compilações, simulações etc.

Quanto aos aspectos de gerência, as primitivas da base de dados e LAGO definem níveis de atuação:

- **administração geral**;
- **administração de grupo**;
- **administração de tarefa**;
- **projetista**;

O **administrador geral**, como o próprio nome diz, gerencia todo o ambiente, realizando cadastramento de usuários, criação de grupos e designação de administradores de grupo.

O **administrador de grupo** realiza tarefas de gerência do grupo e dos contextos ao qual os membros de seu grupo têm acesso. E ele, também, que determina início e fim de transações de projeto (envolvendo o trabalho de um grupo

de projetistas no desenvolvimento de um projeto comum; transação longa).

As funções de **administração de tarefa** são realizadas pelo projetista para gerenciar seu trabalho específico dentro de um projeto em andamento. Correspondem à gerência das transações de usuário (tarefas) e manipulação de objetos para compartilhamento com outros projetistas.

As tarefas de projeto propriamente ditas correspondem à ativação de ferramentas como editores gráficos, compiladores e simuladores.

Quanto à navegação na base de dados, ou seja, às diversas operações de consulta que podem ser realizadas sobre agências, alternativas e versões, LAGO oferece um mecanismo de navegação [GED 88] que permite o caminhamento na estrutura hierárquica da base de dados, a partir de duas formas de representação: **árvore e lista**. A forma **árvore** permite trabalhar num estilo manipulação direta [SHN 87] com apontamento dos elementos (agência, alternativa, versão, ocorrência) para exibição, consulta de atributos, etc. Na forma de **lista**, os objetos são listados em uma "caixa de diálogo", permitindo manipulação.

A navegação se dá em conjuntos de objetos especificados pelo usuário, conforme o nível do banco de dados escolhido: banco de dados público, contexto, banco de dados do projetista e combinação destes.

A figura 3.1 mostra a estrutura global de LAGO.

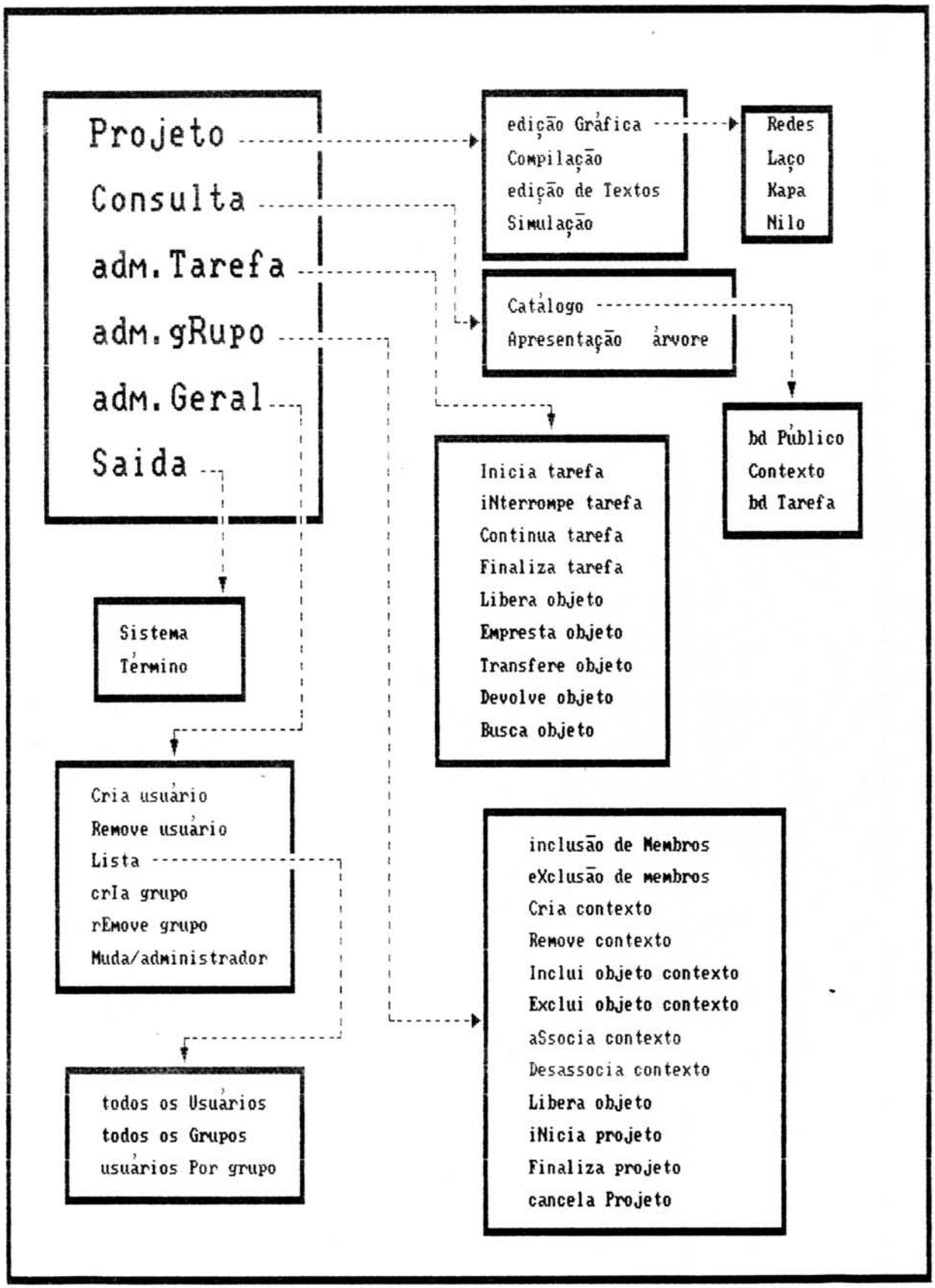


Figura 3.1: Estrutura global de LAGO

3.1.2 Descrição da Interface

A figura 3.2 mostra a tela de apresentação de AMPLO, onde é sobreposta uma caixa de diálogo com a solicitação do código do usuário. Após a informação do código, se este corresponder a um usuário cadastrado no sistema, é solicitada a senha (figura 3.3). Nesta situação, se o usuário quiser alterar sua senha poderá digitar **senha / nova senha** (figura 3.4). Em ambos os casos, LAGO verifica a correção da senha vinculada ao código do usuário. Tendo sucesso nessa fase, o usuário está apto a realizar tarefas no AMPLO, conforme sua classe (projetista, administrador de grupo, administrador geral). E mostrada, então, a tela constante na figura 3.5 com as opções gerais do ambiente e o conjunto de agências da base de dados corrente.

Na **administração geral**, o usuário especial pode cadastrar e remover usuários, pode também, criar, remover e consultar grupos e usuários, e ainda, mudar a administração de grupo (figuras 3.6 a 3.10).

Na **administração de grupo**, o administrador pode alterar grupos, criar e remover contextos, incluir e excluir objetos de contexto, associar e desassociar contextos de grupos, liberar objetos para o banco de dados público, iniciar, finalizar e cancelar transações de projeto (figuras 3.11 e 3.12).

Na **administração de tarefa** (figura 3.12), o usuário pode iniciar, interromper, continuar e finalizar uma tarefa (transação de usuário). Pode, também, manipular objetos (liberar, emprestar, transferir, devolver e buscar objetos).

O **projetista** pode fazer edição gráfica, ou seja,

pode, através dos editores gráficos ou compiladores (NILO, KAPA, LAÇO ou REDES), criar objetos (agências, alternativas e versões), fazer edição de textos, validar o projeto de sistemas digitais, usando o ambiente de simulação (figuras 3.13 e 3.14).

LAGO permite, ainda, saída temporária ao sistema operacional (figura 3.19), como os ambientes atuais de programação. O retorno ao LAGO (programa residente) é efetuado pela função "EXIT" do DOS.

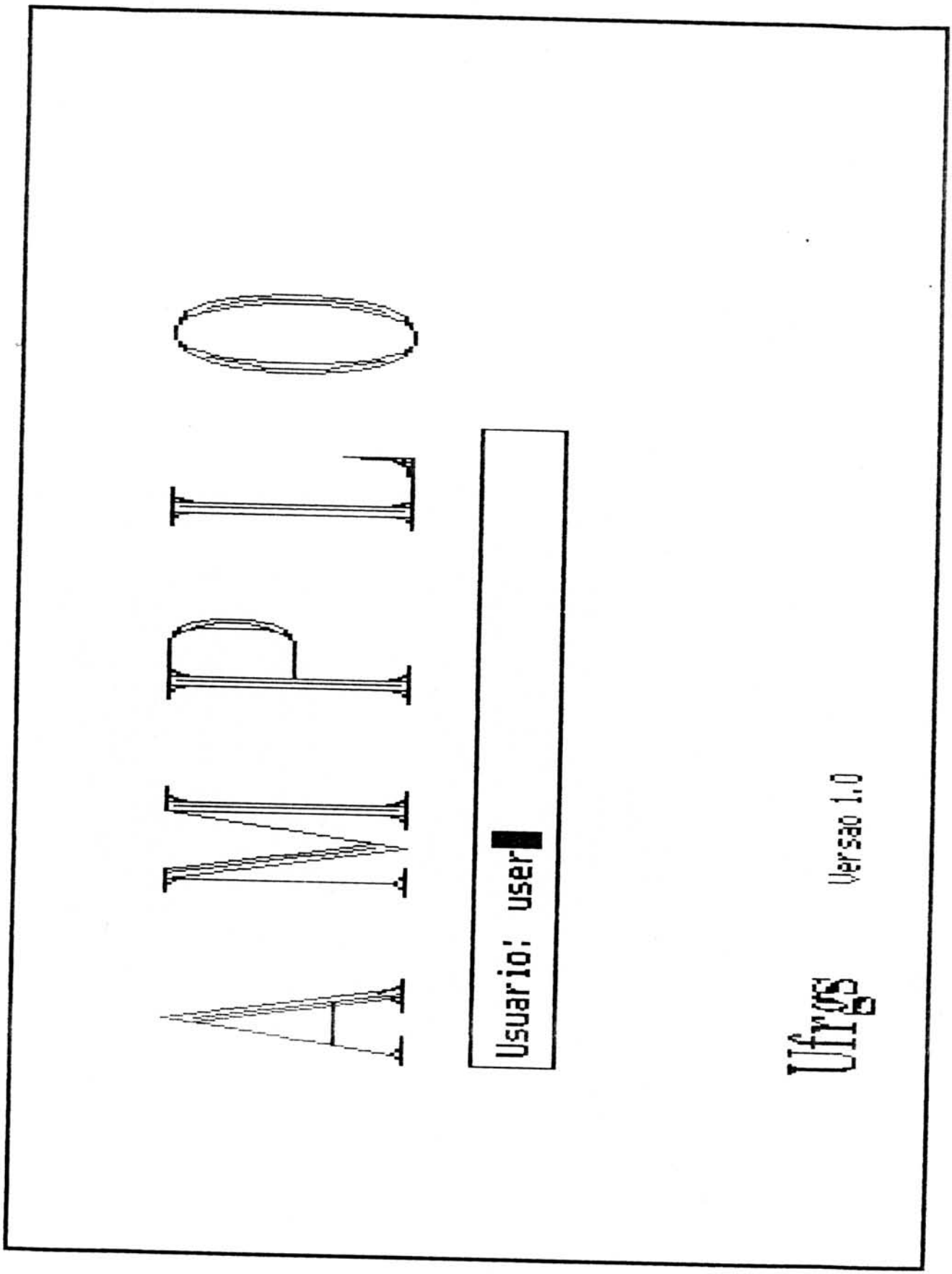


Figura 3.2: Abertura e identificação do usuário

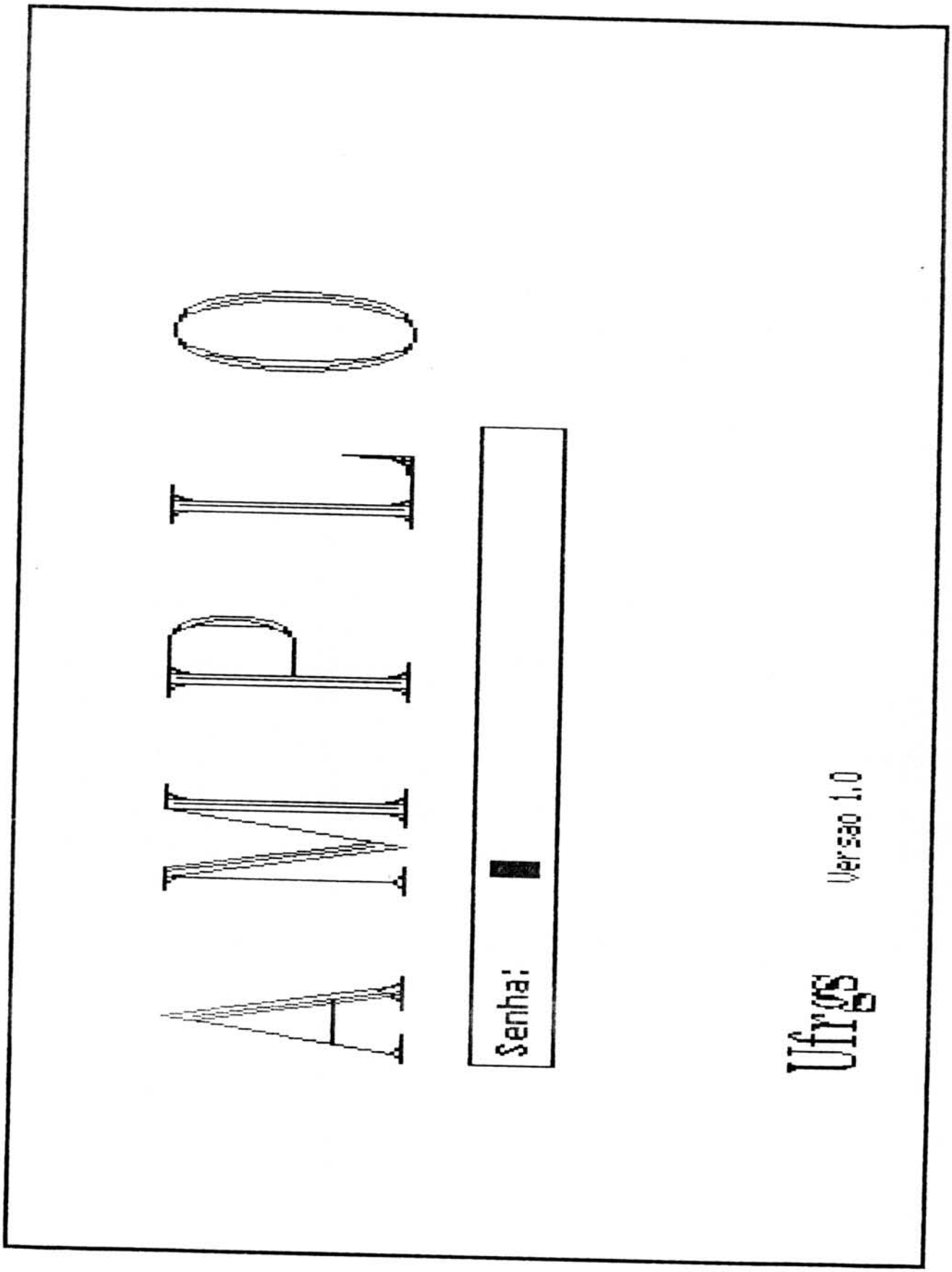


Figura 3.3: Solicitação da senha

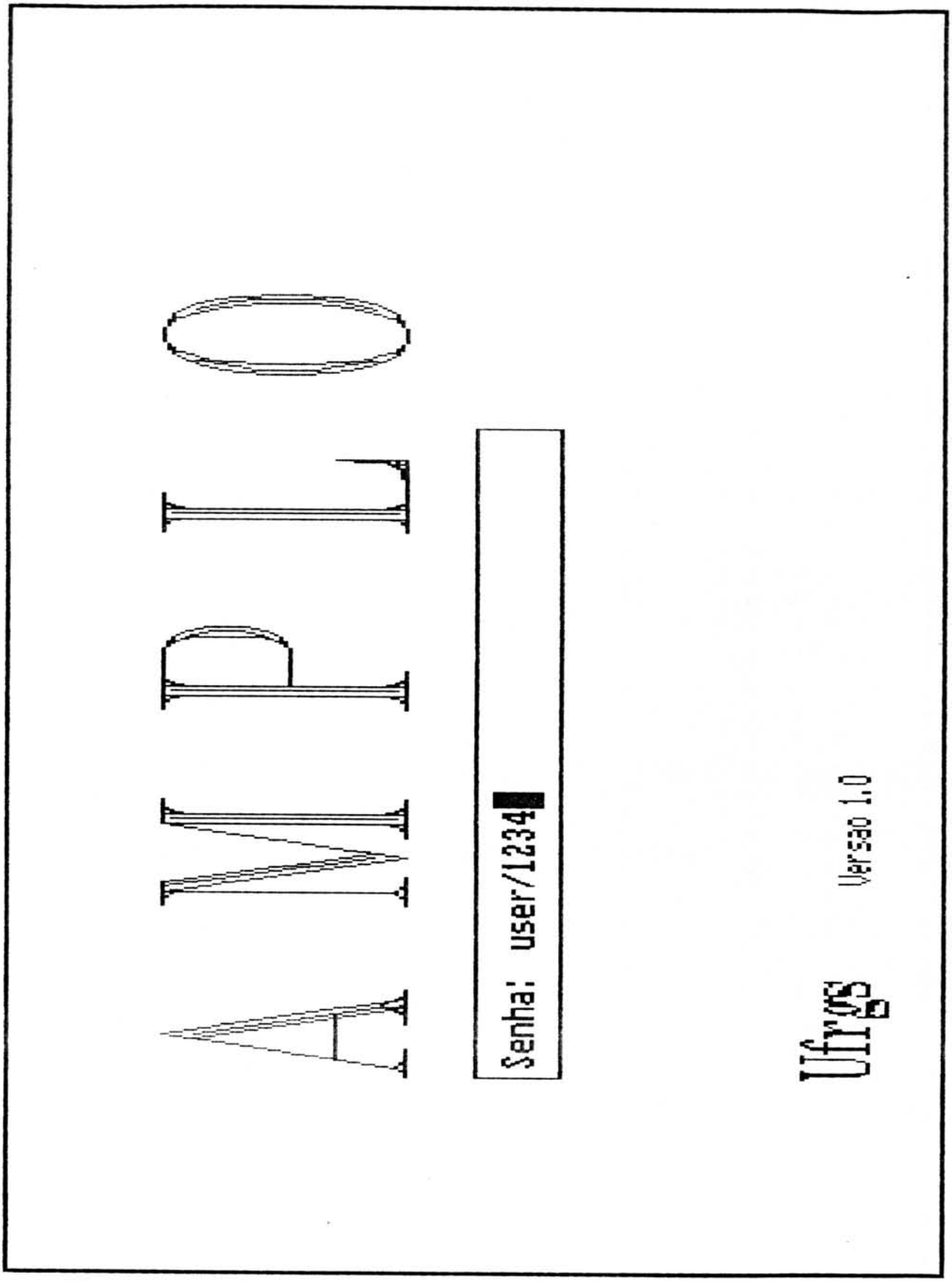


Figura 3.4: Troca de senha

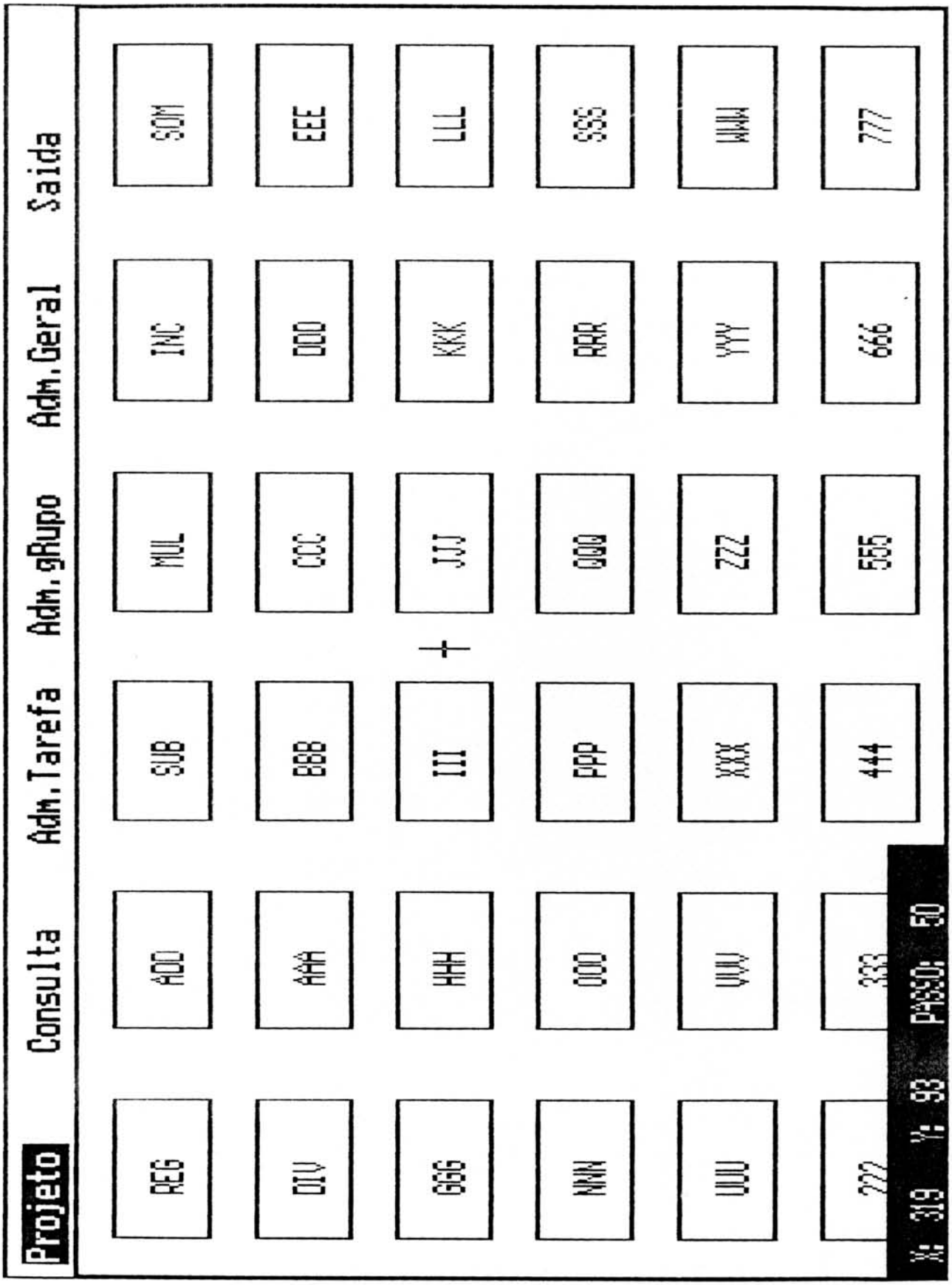


Figura 3.5: Interface LAGO; diretório de agências

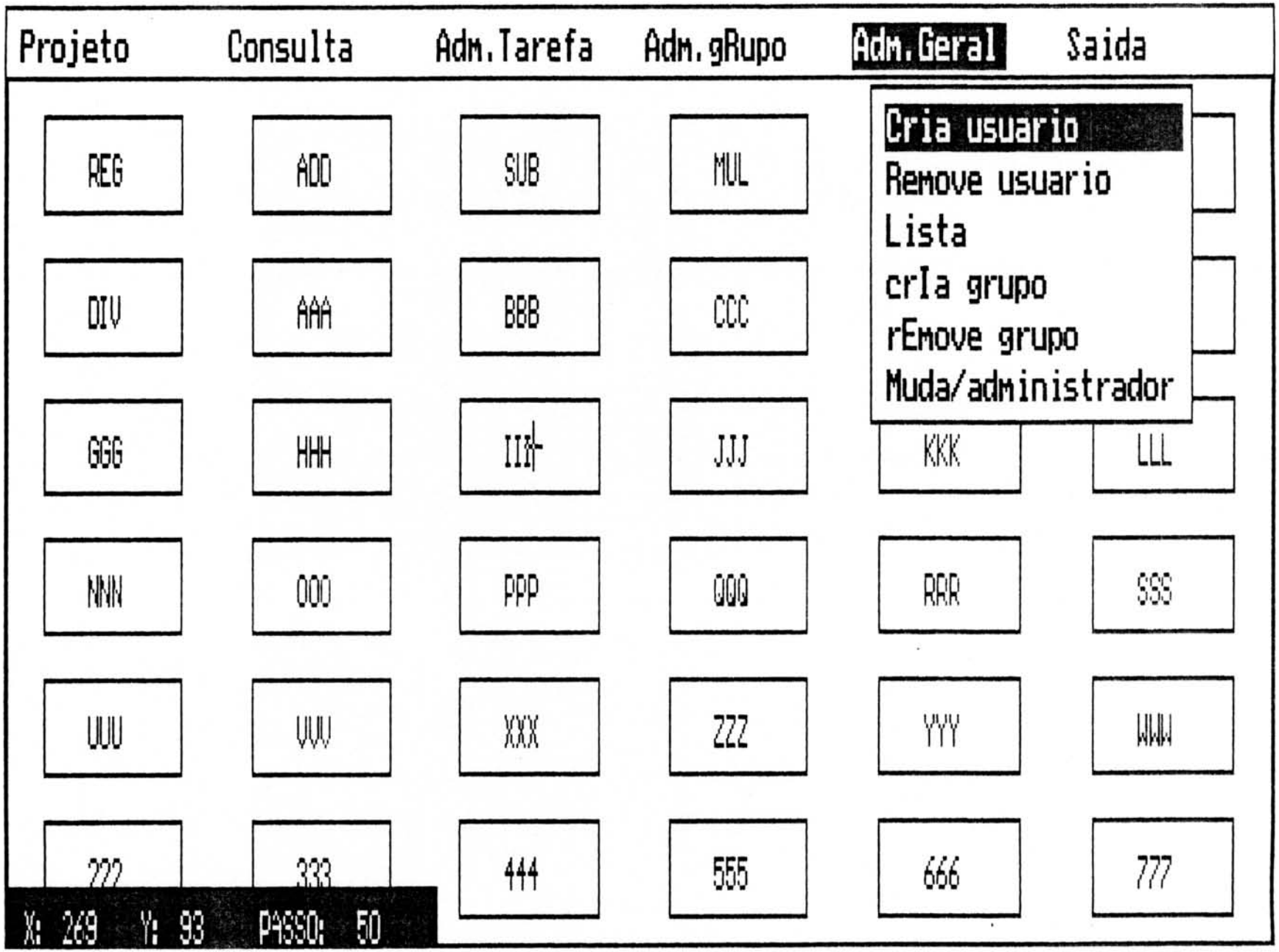


Figura 3.6: Funções de "administração geral"

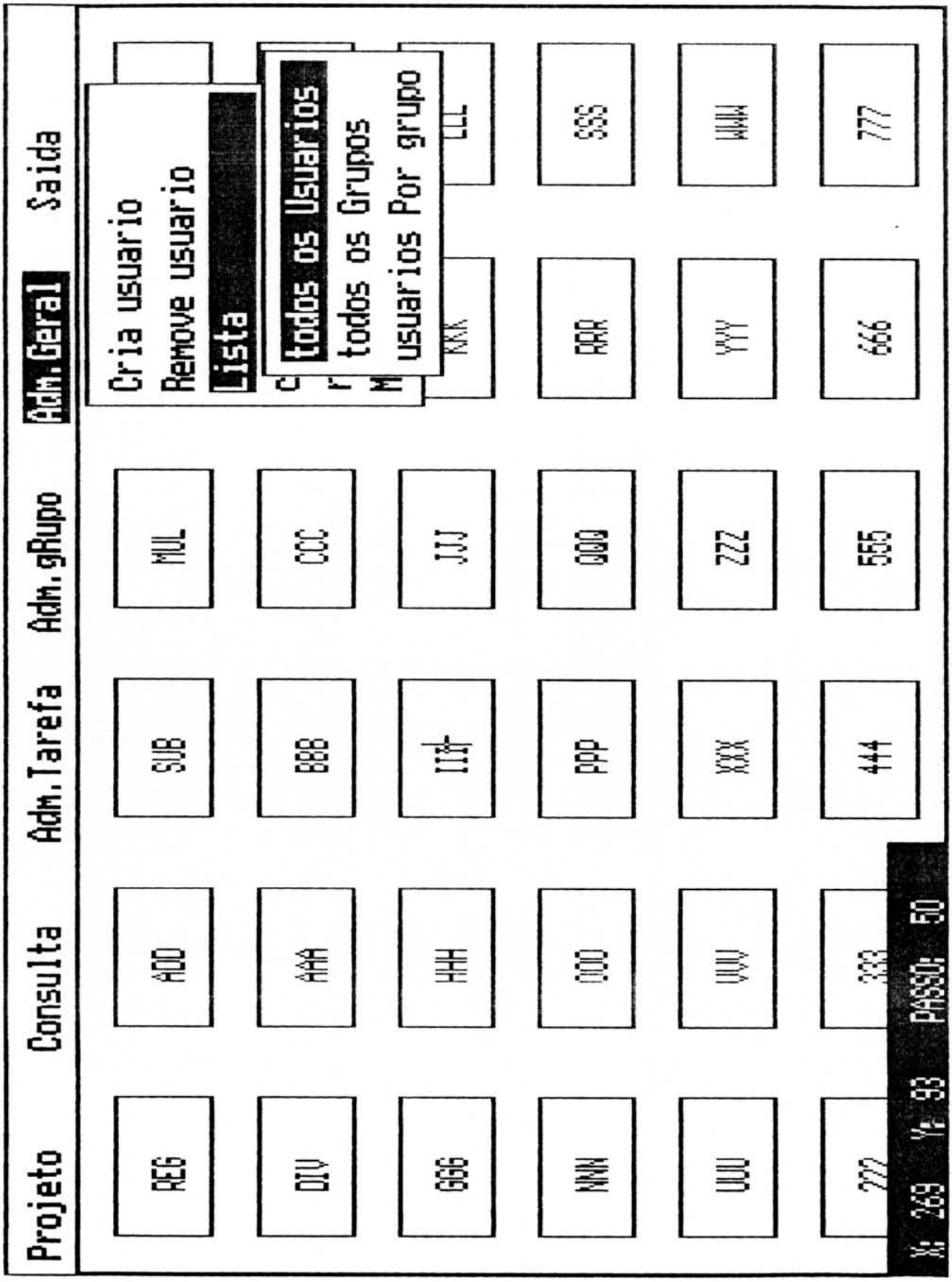


Figura 3.7: Função "lista"

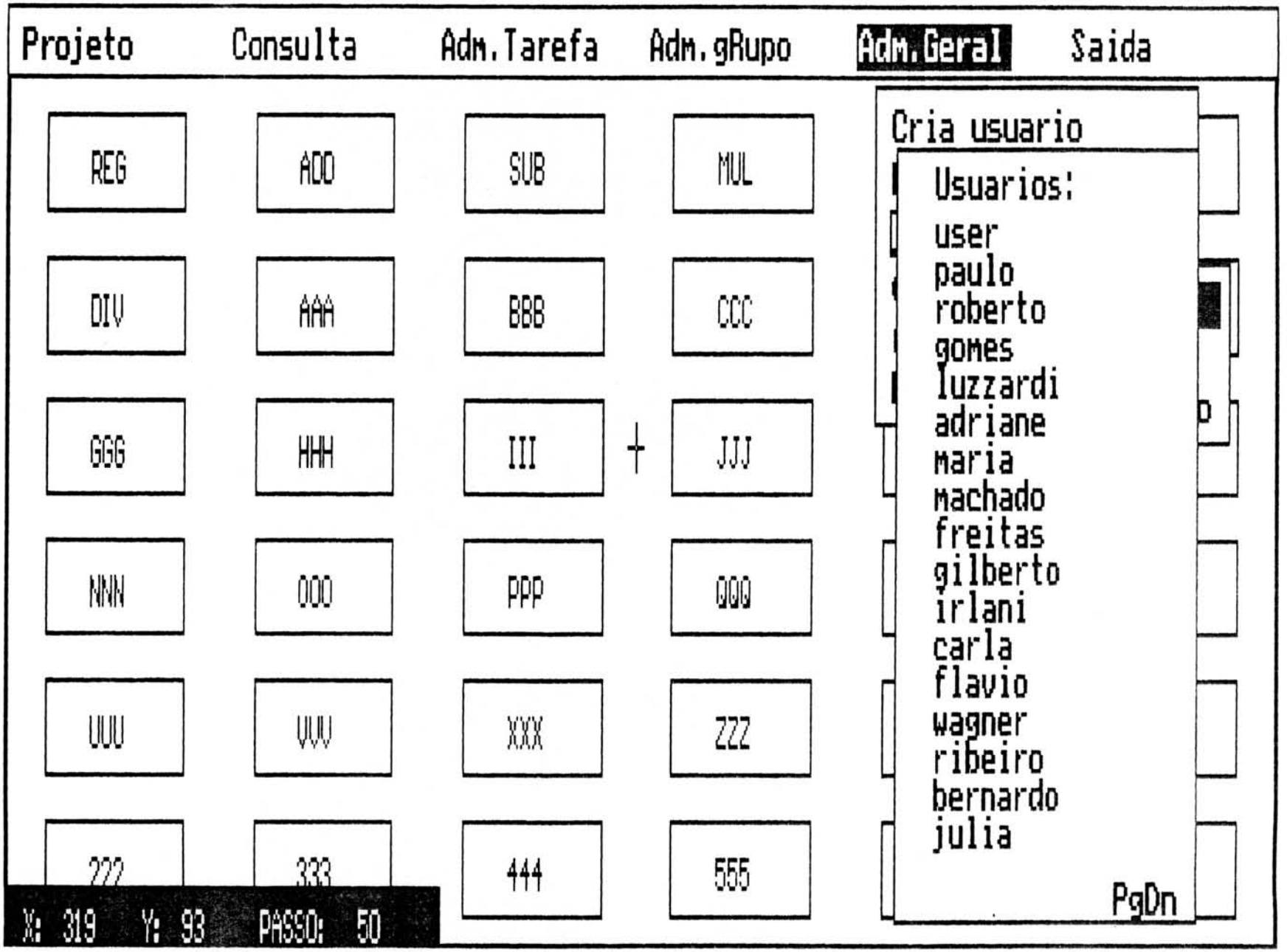


Figura 3.8: Função "lista todos os usuários"

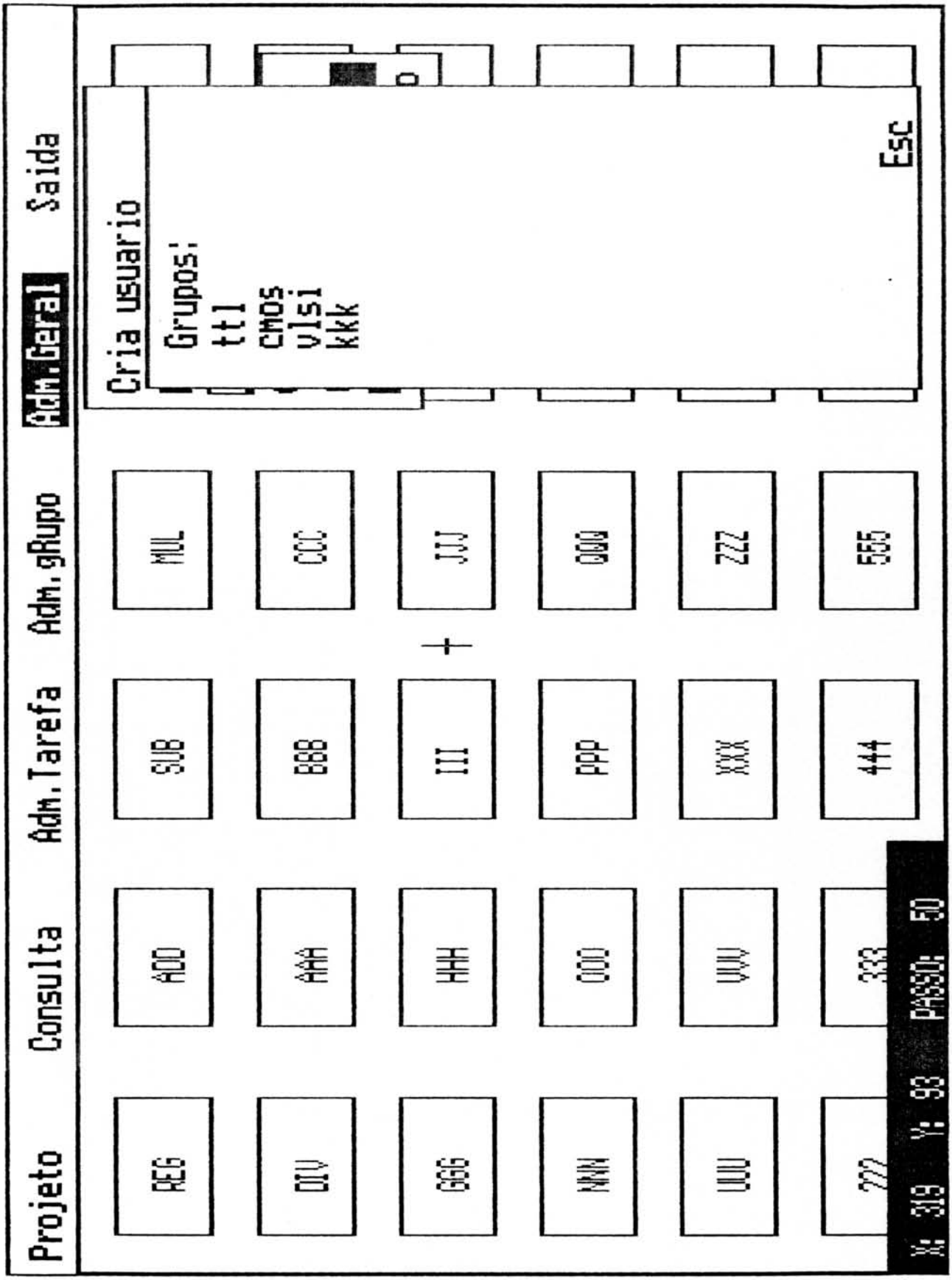


Figura 3.9: Função "lista todos os grupos"

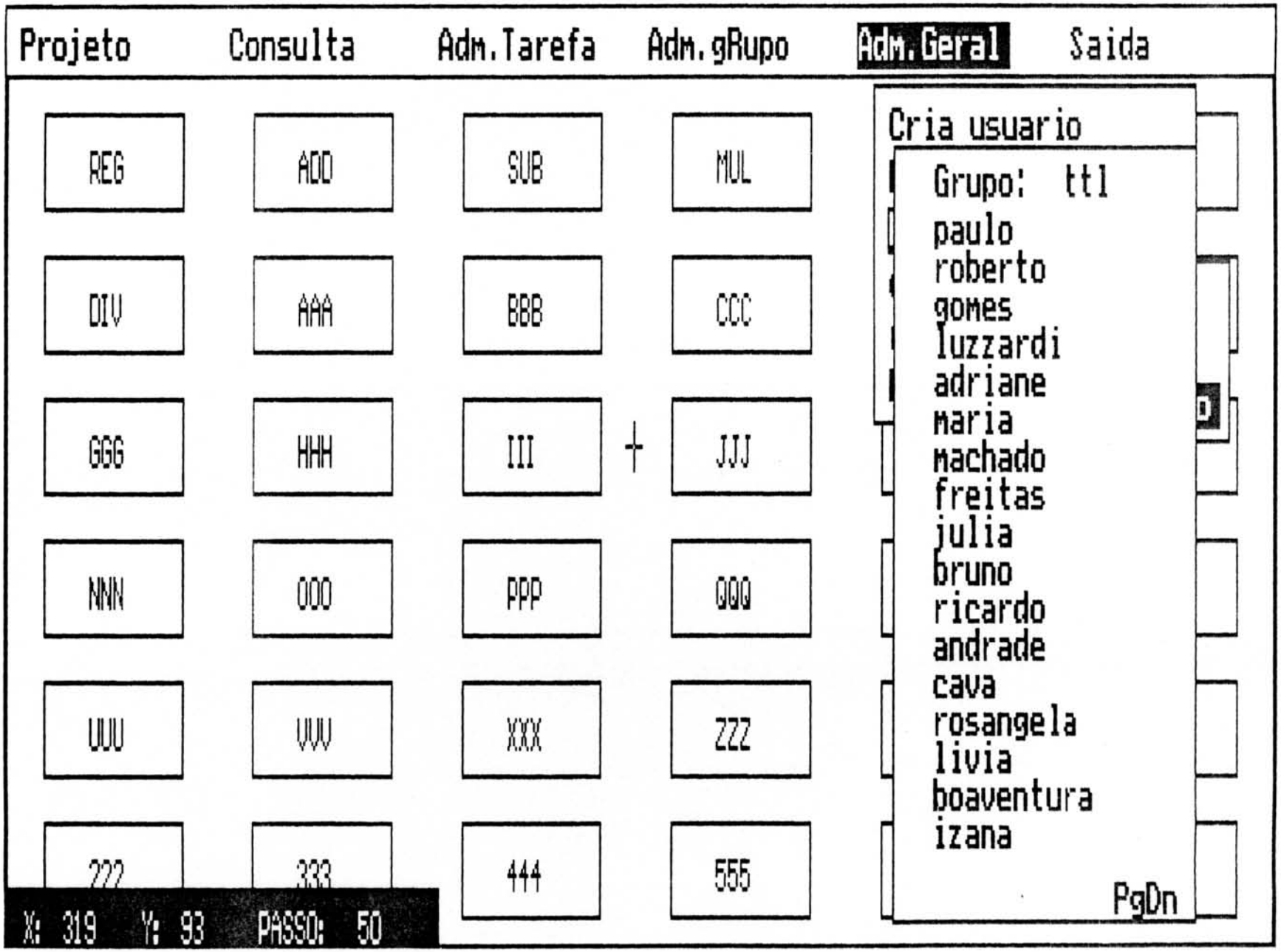


Figura 3.10: Função "lista usuários por grupo"

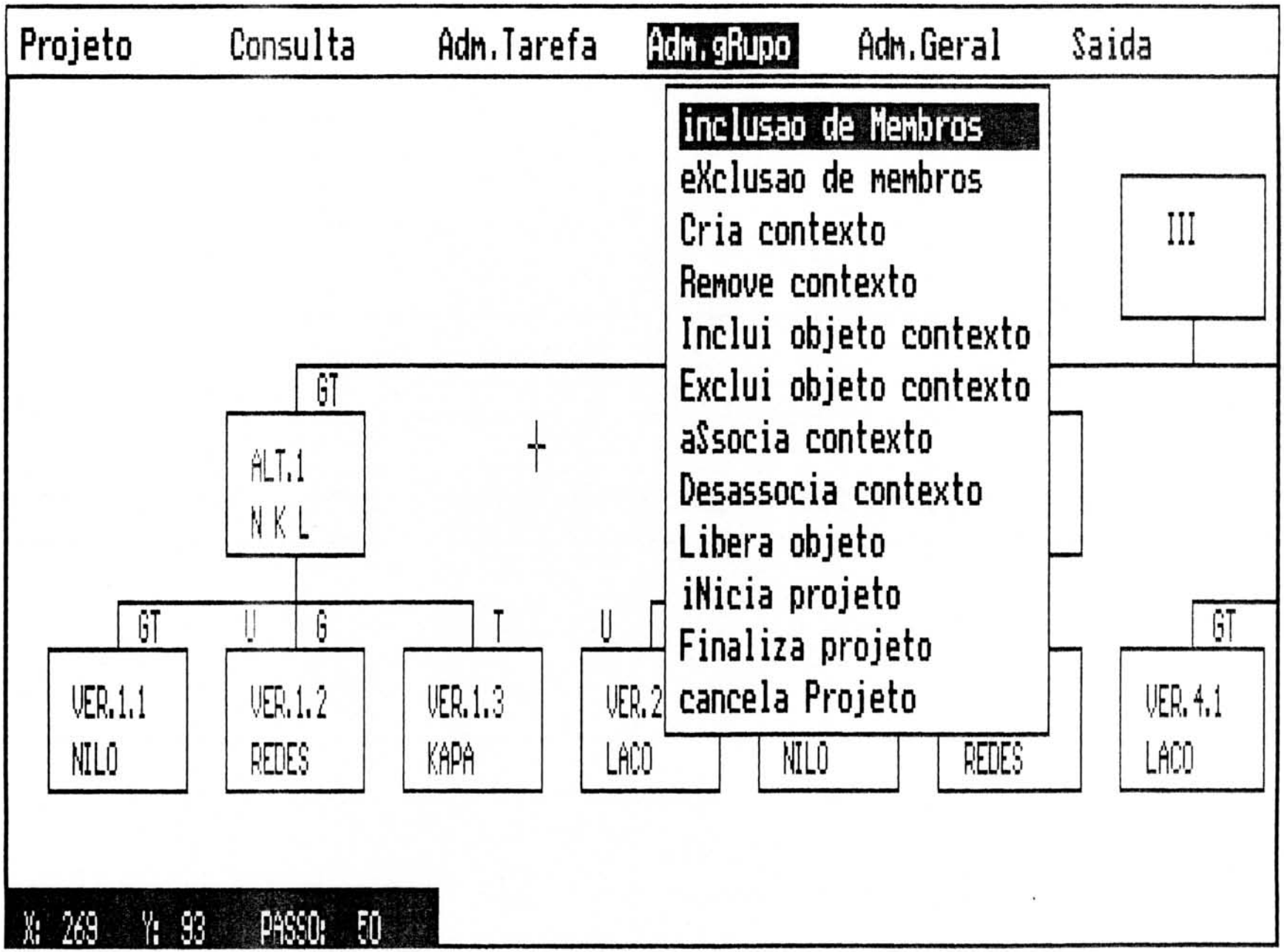


Figura 3.11: Funções de "administração de grupo".

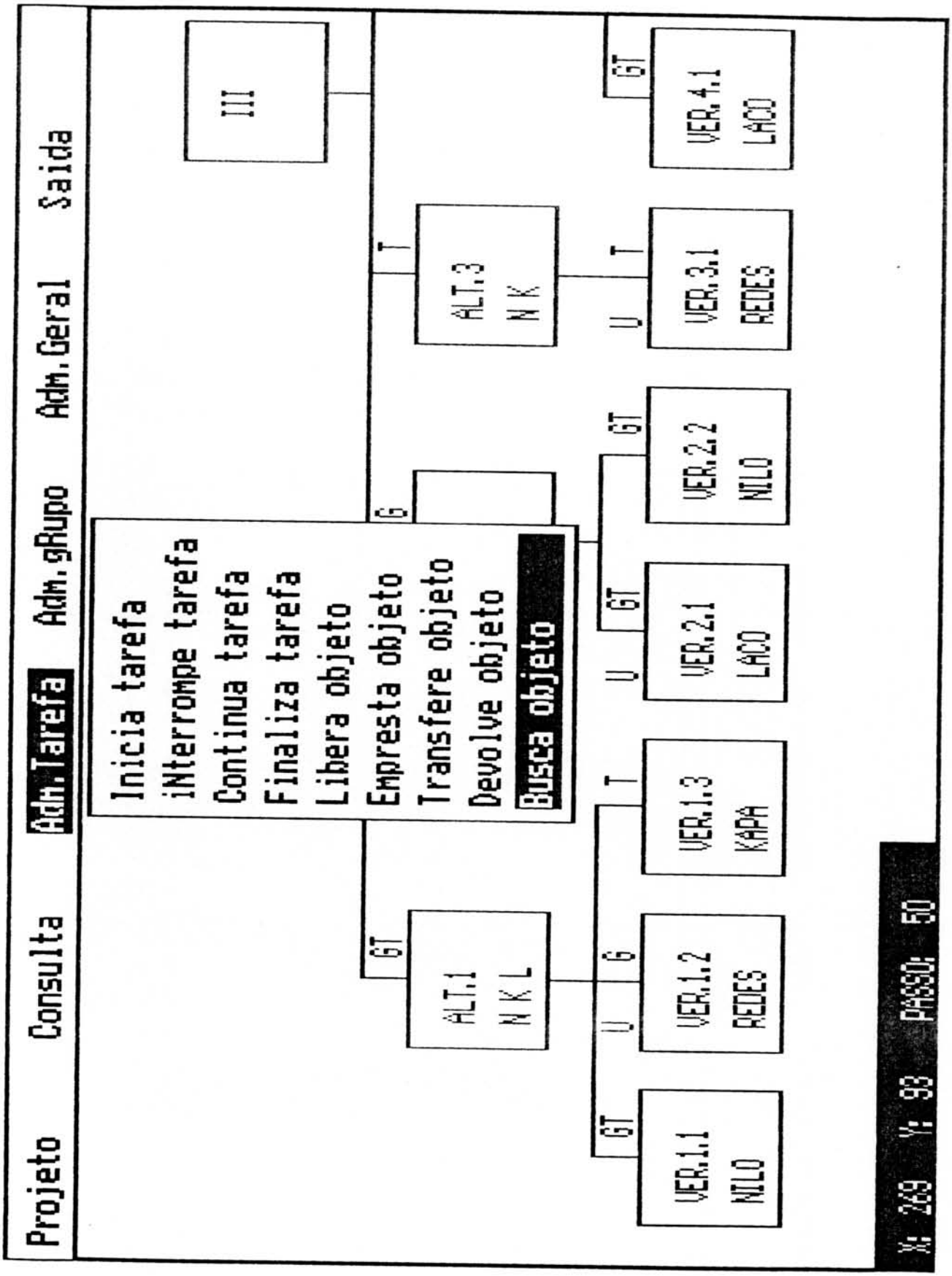


Figura 3.12: Funções de "administração de tarefa"

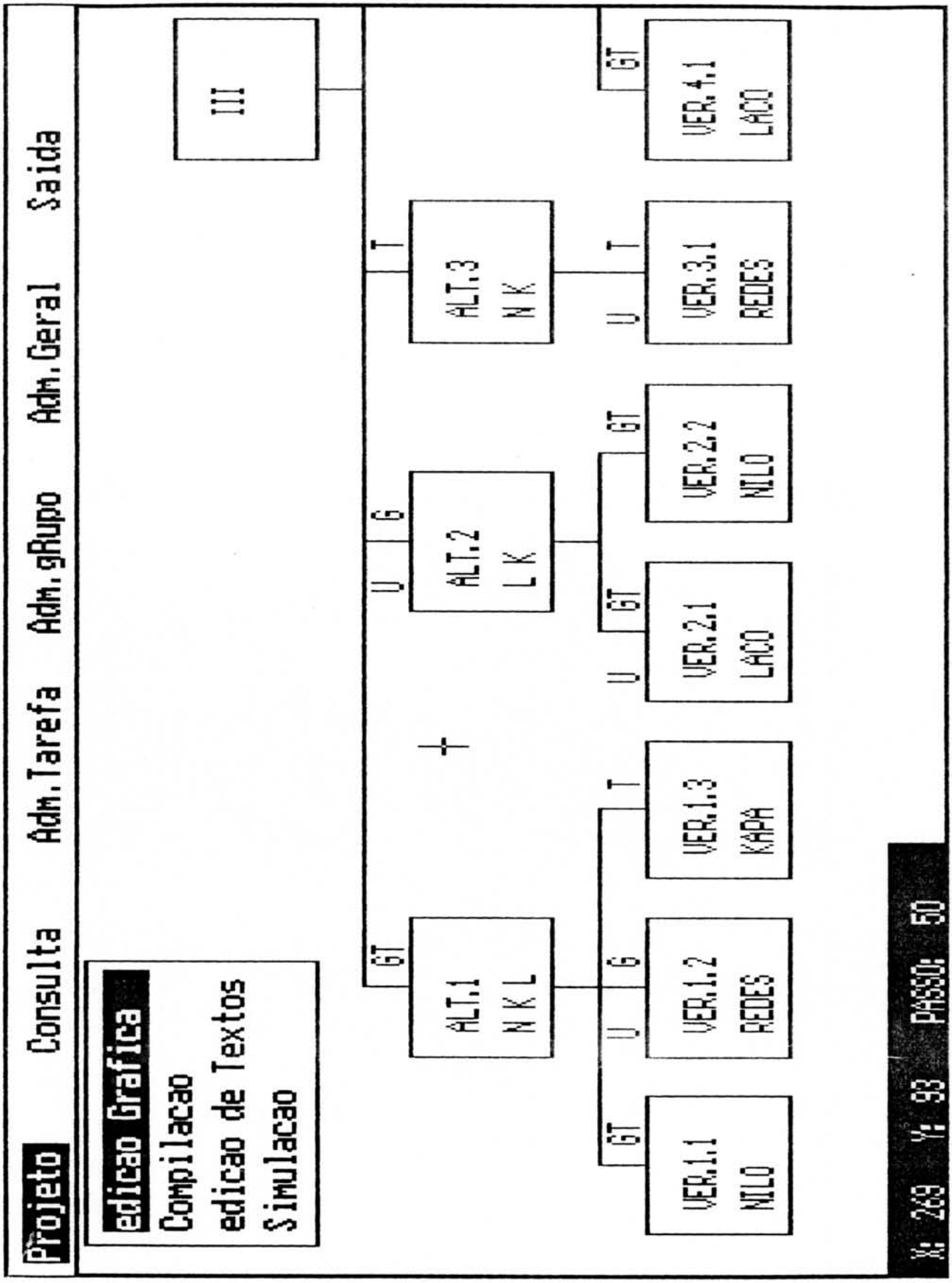


Figura 3.13: Funções de "projeto"

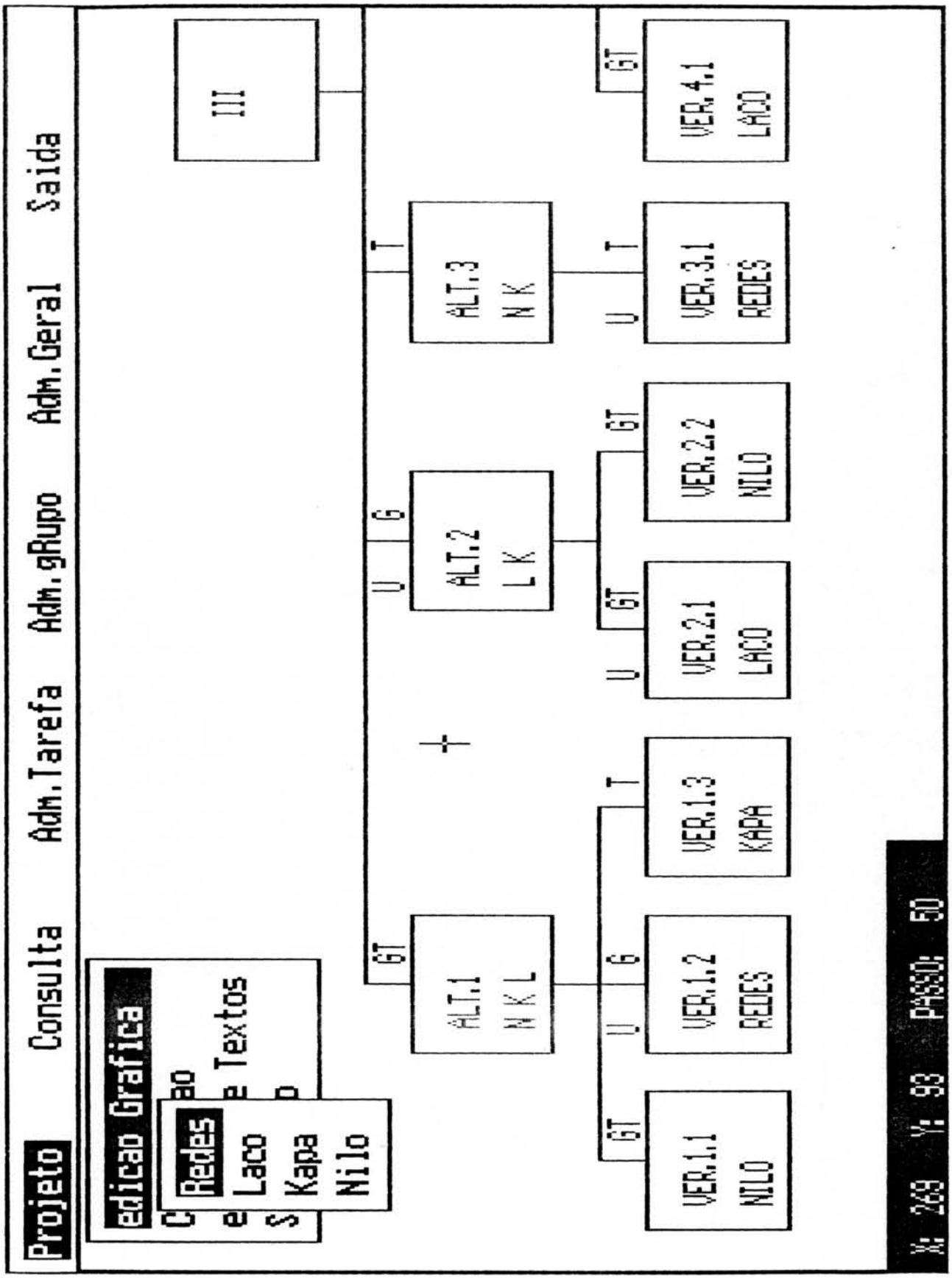


Figura 3.14: "Editores Gráficos"

Inicialmente, quando o sistema é ativado e é exibido o conjunto de opções gerais, é mostrado, também na área de trabalho, o conjunto de agências armazenadas na base de dados privativa do projetista (figura 3.5). A partir deste momento, as funções de navegação estão ativas. Dependendo da opção feita pelo usuário nas funções de **consulta** (figura 3.17) de apresentação, as agências podem ser exibidas graficamente, representadas por retângulos, ou podem ter apenas seus nomes listados (figuras 3.15 e 3.16).

Pode-se também **controlar** (figura 3.17) as funções de navegação, ou seja, selecionar quais as bases de dados (BD público, BD projeto ou BD privativo) a serem consultadas (figura 3.18), qual o tipo de apresentação dos dados (árvore ou lista) ou fazer uma seleção ("pruning" [GED 88]).

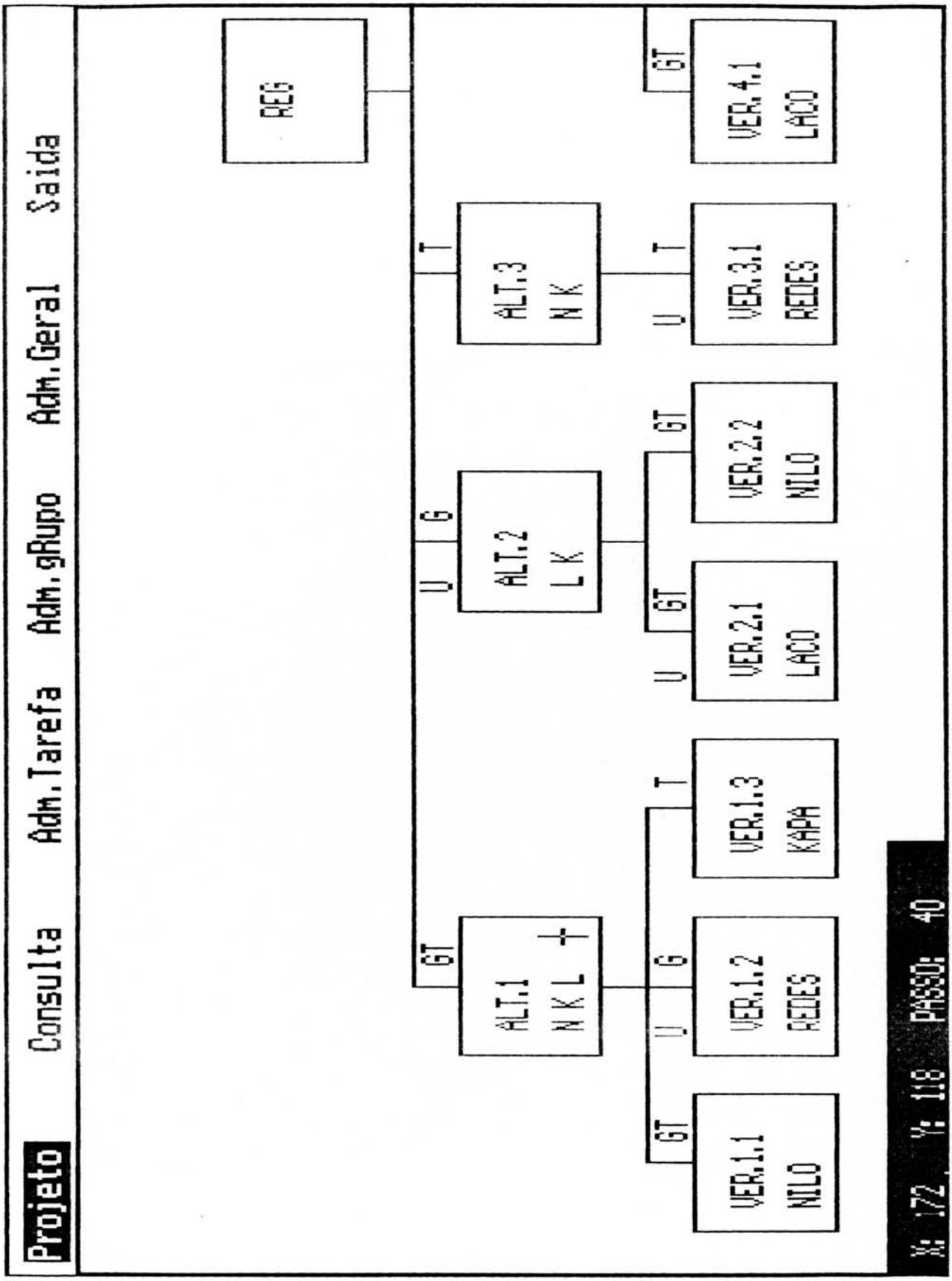


Figura 3.15: Arvore de composição de uma agência

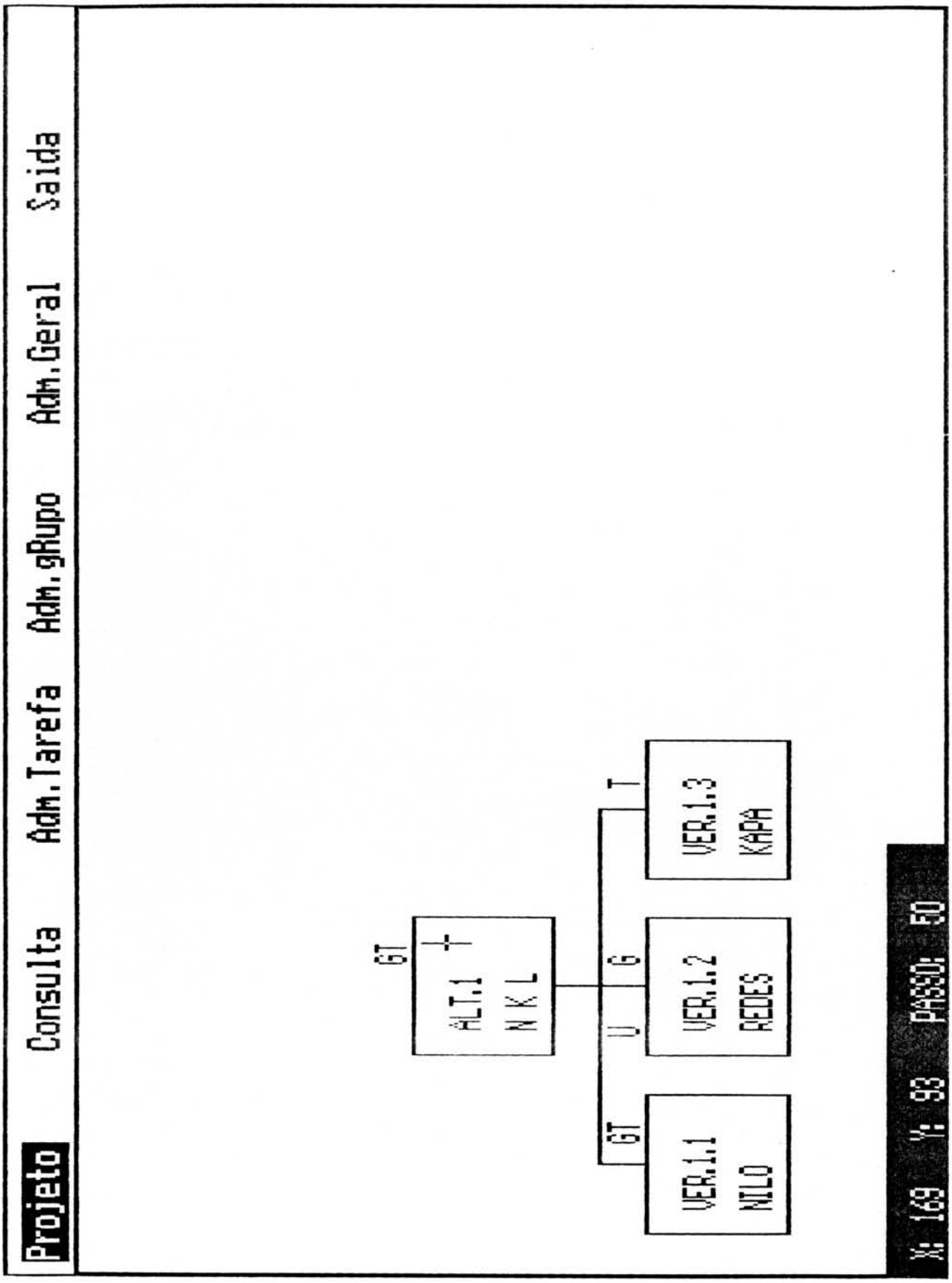


Figura 3.16: Arvore de composição de uma alternativa

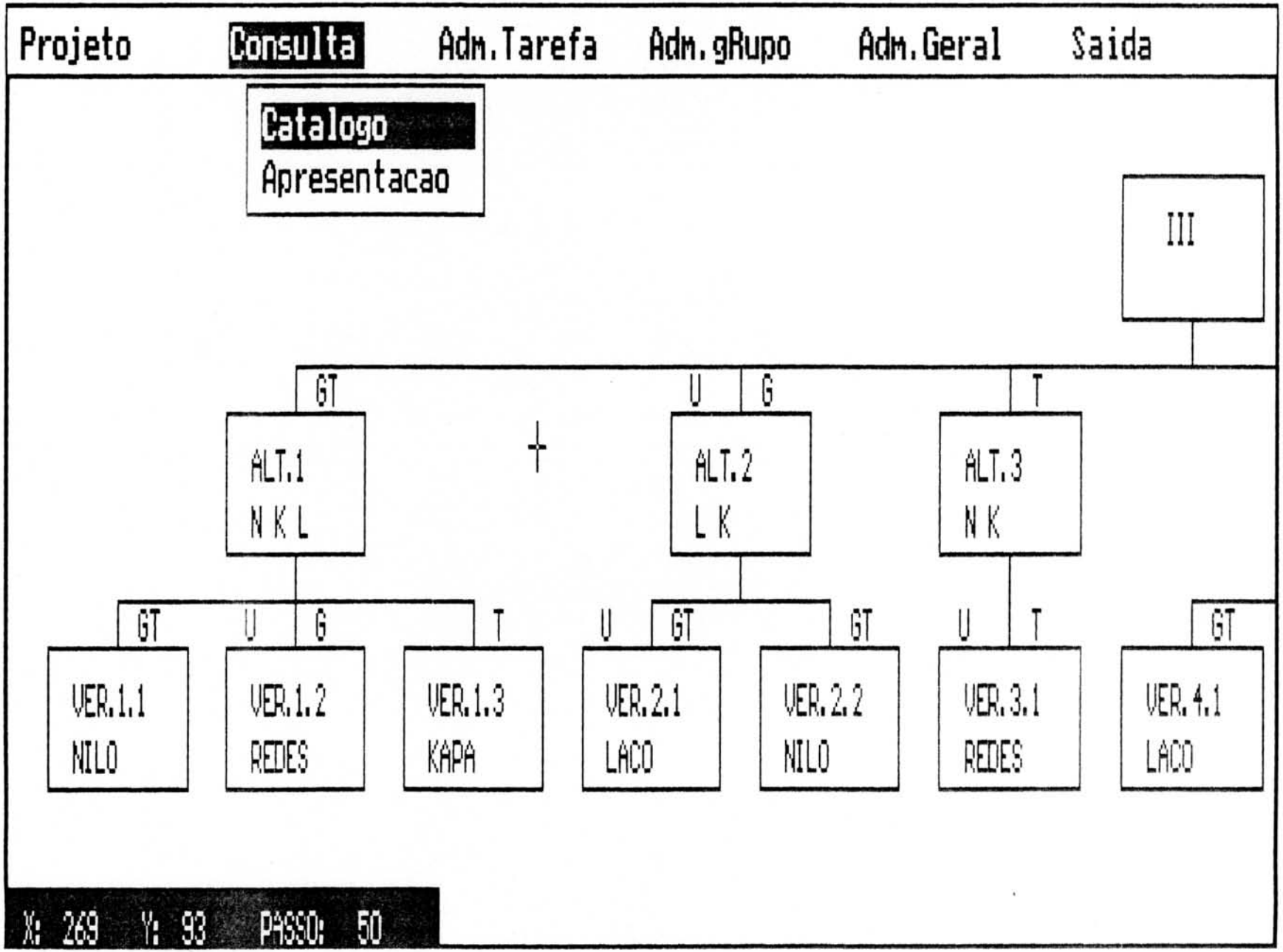


Figura 3.17: Funções de "Consulta".

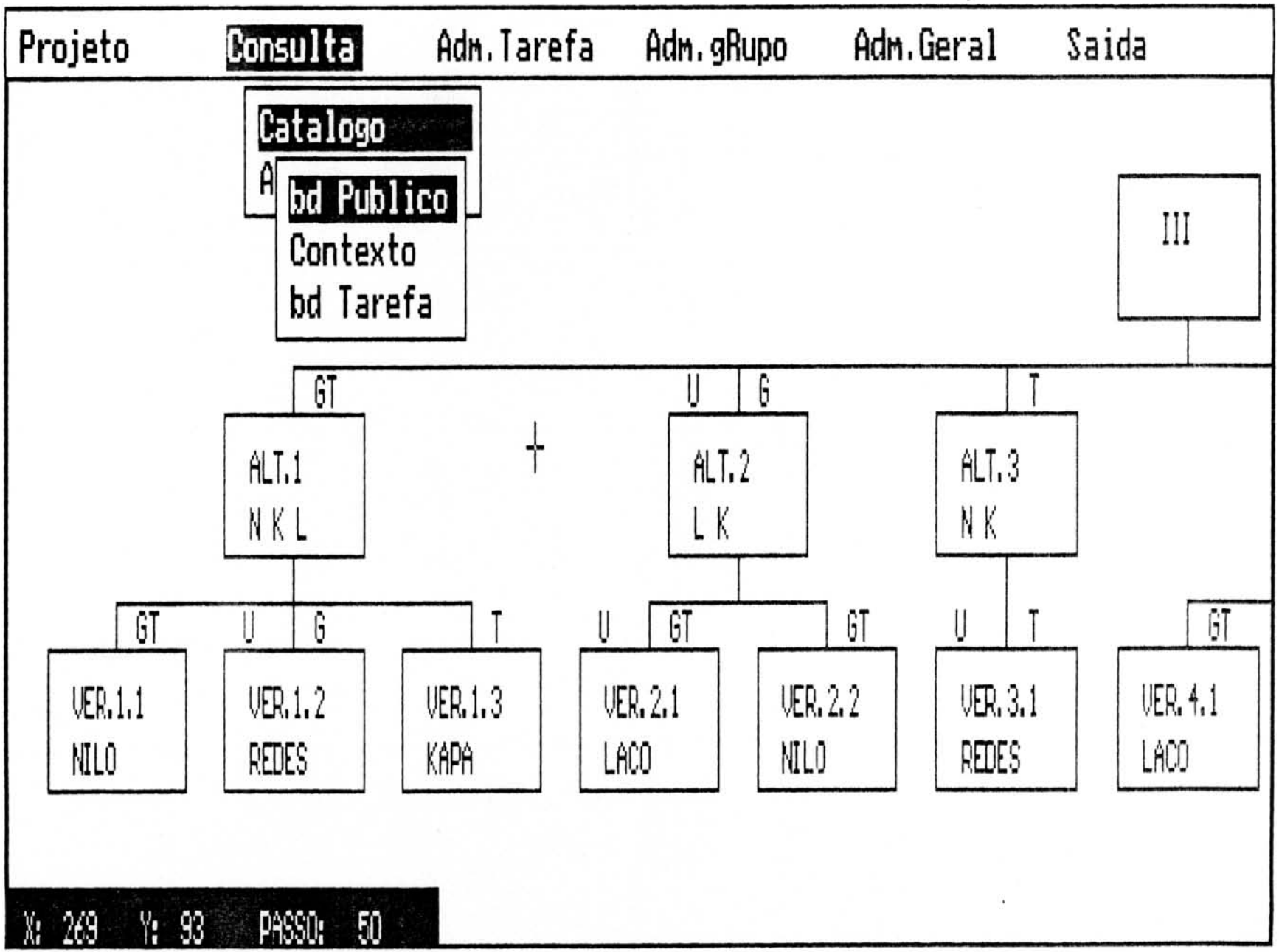


Figura 3.18: Função de troca da base de dados

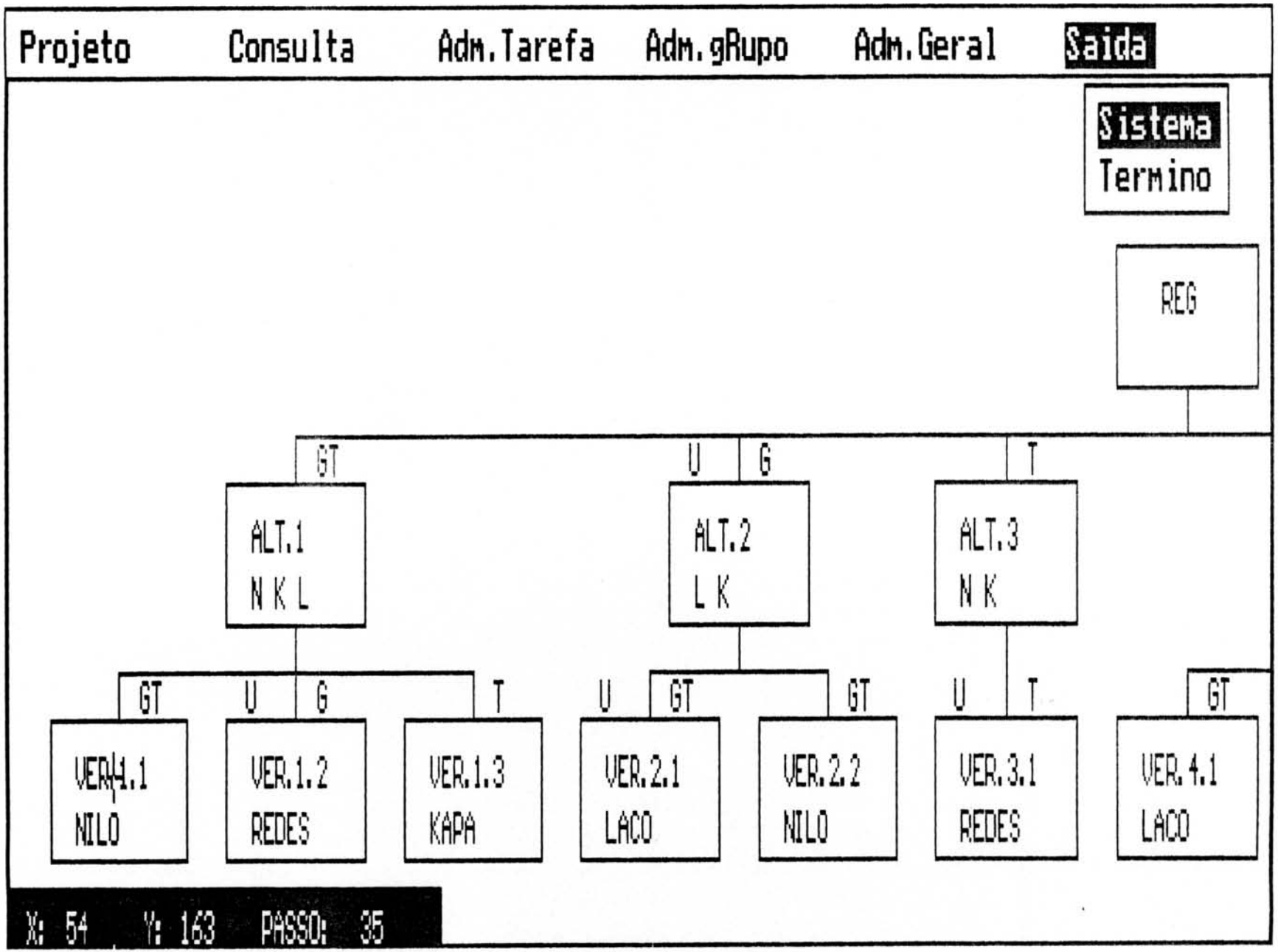


Figura 3.19: Funções de "Saida"

3.2 Acesso ao sistema

Para acessar o ambiente AMPLO (figuras 3.2 e 3.3), o usuário deve se identificar através de código e senha, conforme abaixo:

Usuário: até 12 caracteres;
Senha: até 12 caracteres;

Exemplo:

Usuário: joao <ENTER>
Senha: 1234 <ENTER>

Com estes dados é verificada a existência do usuário. Se o usuário não estiver cadastrado ou a senha não conferir, mensagens de erros são exibidas (figura 3.20):

ERRO_01: Usuário não cadastrado
ERRO_02: Senha inválida

Obtendo sucesso, na entrada do sistema, pelo nome e senha é verificada a prioridade do usuário, a saber:

Prioridade 0: Administração Geral;
Prioridade 1: Administração de Grupo;
Prioridade 2: Administração de Tarefa;
Prioridade 3: Projetista.

Da prioridade do usuário, depende o acesso às opções do **menu principal**:

Adm.Geral: usuários com prioridade 0;
Adm.Grupo: usuários com prioridade 1;
Adm.Tarefa: usuários com prioridade 2;
Demais Opções: usuários com prioridade 0,1 e 2.

A alteração de senha é possível, na entrada do

sistema, quando o usuário especifica sua senha. Ao colocar senha_antiga / senha_nova, o usuário está trocando de senha. É solicitada a re-digitação da nova senha para efeito de confirmação (figura 3.4).

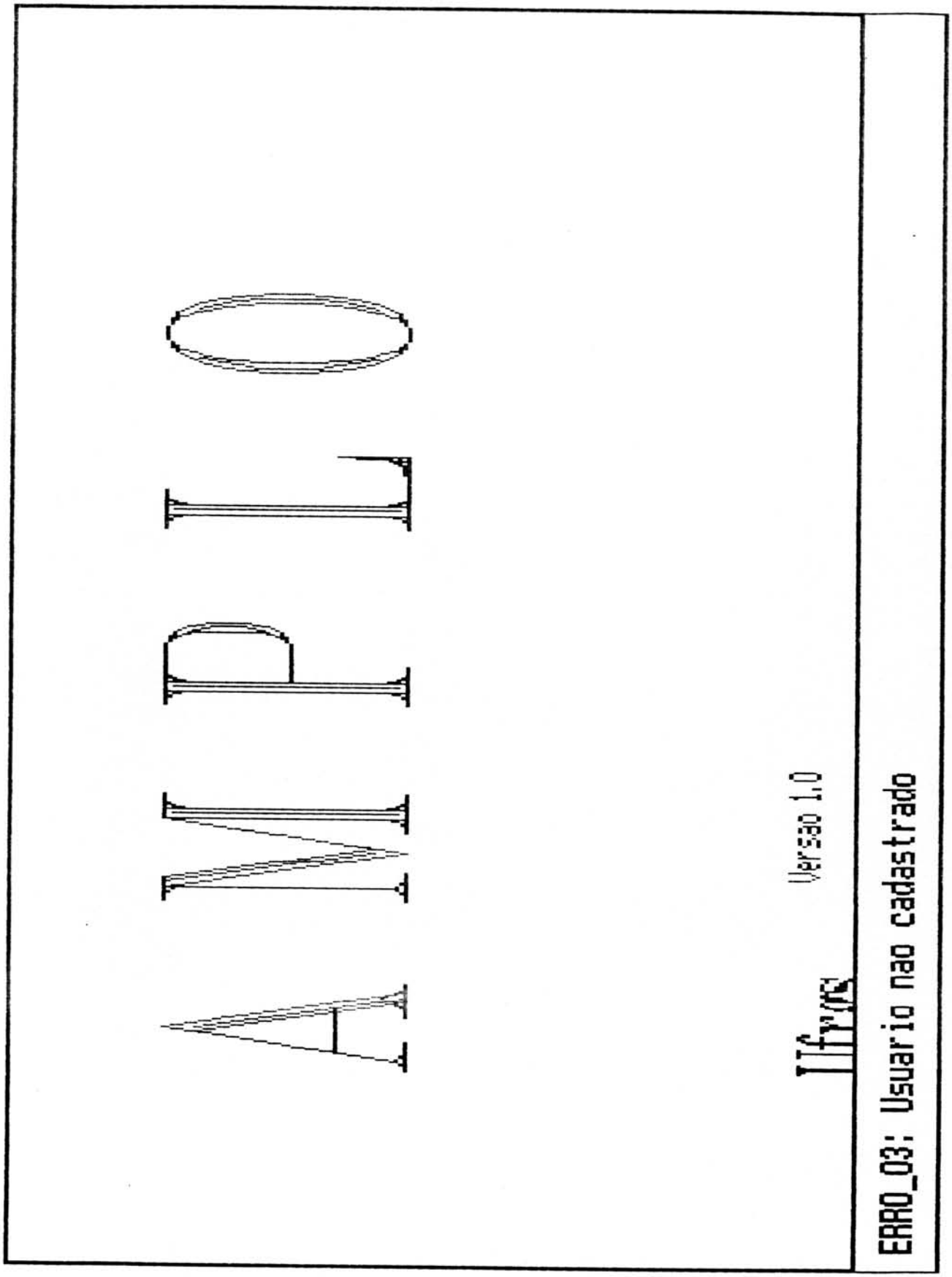


Figura 3.20: Erro na identificação do usuário

3.3 Funções de Gerência

3.3.1 Administração Geral (figura 3.6)

As funções "cria usuário", "remove usuário", "lista ("todos usuários", "usuários por grupo" e "grupos), "cria grupo", "remove grupo" e "muda administrador do grupo" estão previstas na base de dados, mas como ainda não estão disponíveis, foram todas implementadas no decorrer deste trabalho.

Para cadastrar um usuário, o **administrador geral** utiliza a operação **cria Usuário**.

Exemplo:

Usuário: ricardo <ENTER>
Observação: A senha criada internamente é "ricardo".

Para criar grupos através da função **cria grupo**, são fornecidos ao sistema o nome do grupo e o nome do administrador do grupo.

O **administrador geral** pode trocar o administrador do grupo, através da função **mudança de administrador**. São solicitados o **nome do grupo** e o **nome do novo administrador**, e então, ocorre a mudança na administração do grupo.

Para auxiliar o **administrador geral**, existe a função **lista** (figura 3.7 a 3.10), que é constituída de três funções:

- **todos os Usuários**;
- **todos os Grupos**;
- **Usuários por Grupo**.

O **administrador geral** pode listar **todos os usuários**, aparecendo na tela os nomes dos **usuários**. Pode listar **todos os grupos**, aparecendo na tela a identificação de cada **grupo**, ou pode listar os usuários que pertençam ao grupo (**usuários por grupo**), aparecendo na tela os códigos dos **usuários**.

O **administrador geral** pode remover usuários, através da função **remove usuário**. O usuário é removido do banco de dados. Pode-se também remover o grupo, através da função **remove grupo**. O grupo passa a não existir, mas os usuários permanecem tendo acesso ao sistema.

3.3.2 Administração de Grupo (figura 3.11)

As funções de administração de grupo permitem a formação efetiva do grupo e a gerência de certos aspectos de projeto, particularmente em relação aos objetos aos quais o grupo tem acesso.

As funções "inclusão de membros" e "exclusão de membros" estão previstas na base de dados, mas como ainda não estão disponíveis, foram todas implementadas no decorrer deste trabalho.

Na **inclusão de membros** são solicitados o **nome do grupo** e o **nome do usuário**; na **exclusão de membros** são solicitados o **nome do grupo** e o **nome do usuário** que será excluído deste grupo. Pode-se excluir quantos usuários se desejar, inclusive permanecendo o grupo vazio (só com o administrador do grupo). Pode-se, também, criar um contexto, definindo-o a partir de regras ou enumerando os objetos contidos no mesmo. Contextos são associados a grupos que passam a ter o direito de realizar projetos usando determinados contextos. Objetos podem ser incluídos ou excluídos de contextos. Um contexto também pode ser removido; a única restrição é que os projetos dos grupos associados a ele tenham sido encerrados, não utilizando mais seus objetos.

O projeto pode ser cancelado ou finalizado. Neste segundo caso, todas as tarefas devem ter sido encerradas e todos os objetos liberados do contexto para o banco de dados público ou removidos.

3.3.3 **Administração de Tarefa** (figura 3.12)

Ao começar uma parte do projeto ao qual está vinculado, o projetista notifica o sistema que vai iniciar uma tarefa. O sistema verifica se não existe outra tarefa com mesmo nome. É criado um banco de dados privativo. O projetista pode interromper uma tarefa, marcando um "savepoint" no "log" do banco de dados privativo. Posteriormente a tarefa pode continuar.

O projetista pode emprestar objetos (autorizar o empréstimo) a outro projetista. O objeto não pode ser reemprestado. A real transferência só se dará após um **busca objeto**.

O projetista pode **transferir** objetos para o banco de dados privativo de outro projetista. O objeto pode ser transferido ou emprestado novamente, e não precisa ser devolvido. A real transferência só se dará após um **busca objeto**.

O projetista deve devolver o objeto ao banco de dados de origem, através de um **devolve objeto**; só se pode devolver um objeto que foi emprestado.

A real transferência de um objeto que foi emprestado, devolvido ou transferido ocorre através de um **busca objeto**.

O objeto para ser consolidado deve ser liberado, através de um **libera objeto**, passando do banco de dados do projetista para o contexto.

Para finalizar uma tarefa, as seguintes condições devem ser obedecidas: todos os bloqueios (um objeto só pode ser liberado se fizer referência a objetos já liberados) aos objetos consolidados devem ter sido liberados; todos os

objetos criados ou alterados devem ser liberados ou removidos; não pode haver nenhum objeto emprestado, sem que tenha sido devolvido para o banco de dados de origem.

3.4 Funções de Projeto (figura 3.13)

As funções de projeto correspondem à ativação das ferramentas de especificação e de simulação.

As ferramentas permitem 2 tipos de especificação:

- **textual**: editor de textos e compiladores de linguagens de descrição de hardware;

- **gráfica**: editores gráficos, específicos por linguagem.

A função de simulação dá acesso ao ambiente de simulação, dentro do qual podem ser ativados diferentes simuladores (específicos por linguagem de descrição de hardware) e ferramentas diversas para suporte ao processo de simulação.

3.5 Funções de Consulta

As funções de consulta com navegação permitem visualizar a hierarquia de agências presente na base de dados em diferentes níveis de detalhamento.

Inicialmente, quando o sistema é ativado e é exibido o conjunto de opções gerais, é mostrado, também na área de trabalho, o conjunto de agências armazenadas na base de dados privativa do projetista (figura 3.5). A partir deste momento, as funções de navegação estão ativas. Dependendo da opção feita pelo usuário nas funções de **consulta** (figura 3.17), as agências podem ser exibidas graficamente, representadas por retângulos, ou podem ter apenas seus nomes listados (figuras 3.15 e 3.16), dependendo do tipo de apresentação.

Em ambos os casos, as representações das agências ficam sensíveis e, se uma for apontada será trazida da base de dados a árvore (lista) de alternativas da agência indicada. Junto com a árvore de alternativas vêm os seus atributos (nível de descrição, descrição gráfica e/ou textual, indicativo de última alternativa) e as versões associadas (figura 3.21).

O usuário poderia indicar, então, a alternativa e visualizar sua estrutura hierárquica, ou ainda, indicando uma versão, obter a sua árvore de composição se for o caso. Entretanto, o usuário pode necessitar informações a respeito da interface de uma alternativa ou do "conteúdo" de uma versão. Uma dupla indicação ("duplo click") é utilizada e, sendo assim, se for apontada uma **Alternativa**, será exibida sua interface, e se for apontada uma **Versão** (primitiva ou composta), será exibida sua representação dependendo da linguagem (NILO, KAPA, LAÇO ou REDES).

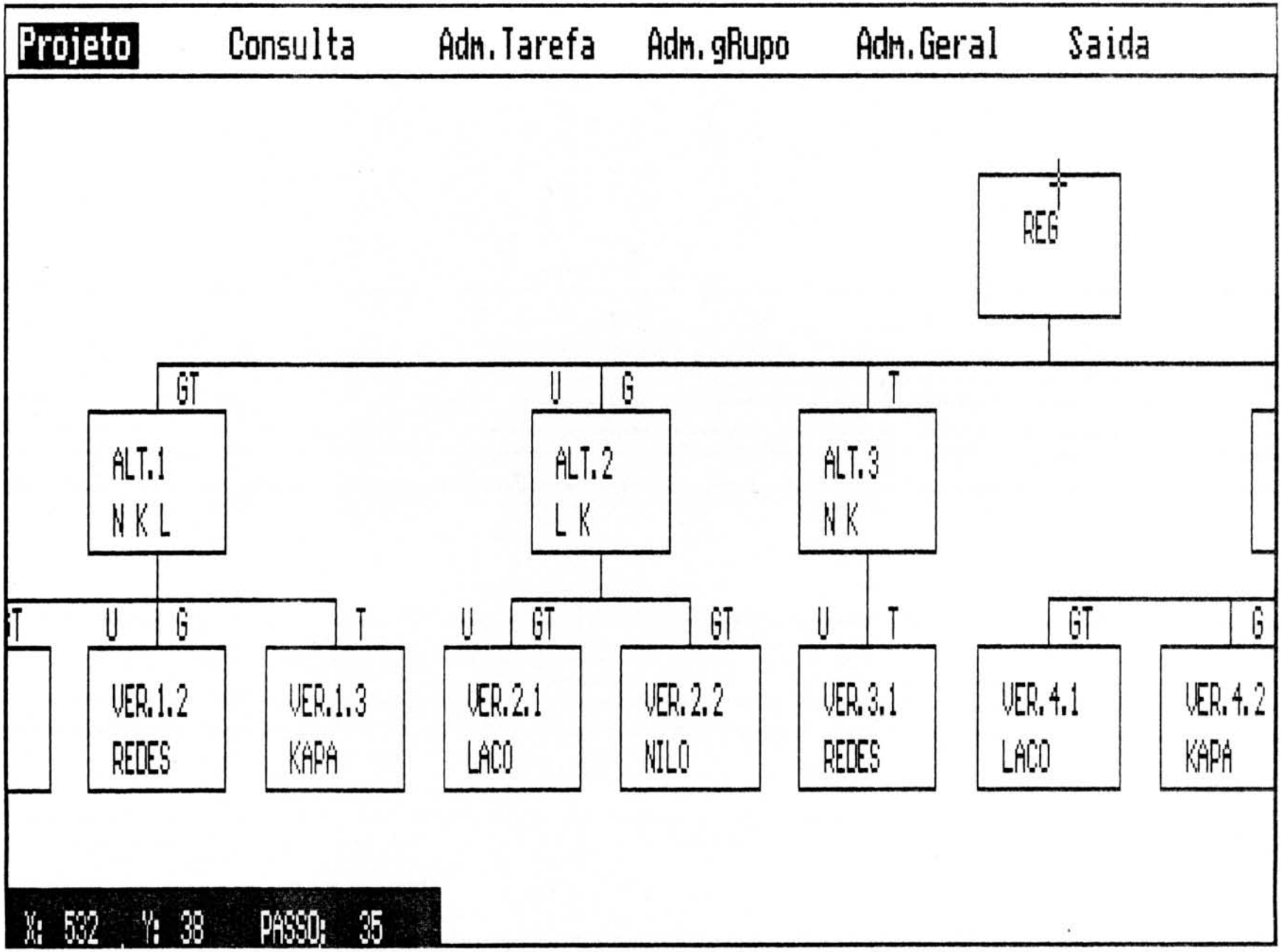


Figura 3.21: Exemplo de uma "agência"

Na situação mostrada na figura 3.21, se apontarmos uma **Agência**, iremos para funções de "poda", ou seja, exibição seletiva (figura 3.22).

Nesta, podem ser exibidas, **todas** as alternativas, alternativas por **nível**, alternativas por **representação** (gráfica ou textual), **última alternativa absoluta** ou alternativa **última por nível**.

Se for apontada uma alternativa, na figura 3.21, são habilitadas as opções de "poda" de alternativa (figura 3.23). Pode-se visualizar todas as versões da alternativa e as versões nas quais ocorrências da alternativa são usadas, através da função **usos**. Pode-se visualizar todas as versões da alternativa, dependendo da linguagem escolhida (NILO, KAPA, LAÇO ou REDES), através da função **versões por linguagem**. Pode-se visualizar todas as versões da alternativa, dependendo da representação (gráfica ou textual), através da função **versões por representação**. Pode ser exibida a última das versões da alternativa apontada, através da função **última versão absoluta**, e a última versão de cada nível da alternativa apontada, dependendo do nível (NILO, KAPA ou LAÇO), através da função **última versão por nível**.

Se for apontada uma **versão**, na figura 3.21, as funções de "poda" (figura 3.24) permitem: exibir as ocorrências (figura 3.25) contidas na versão, se a versão apontada for composta, através da função **composição**; e exibir as versões REDES nas quais ocorrências da versão são usadas, através da função **usos** (figura 3.26). Se a versão for composta, é permitido ainda: exibir a árvore de composição (figura 3.26), dependendo da linguagem (NILO, KAPA, LAÇO ou REDES), através da função **composição por linguagem**; exibir a árvore de composição, dependendo da representação (gráfica ou textual), através da função **composição por representação**; exibir ocorrências de

alternativas ou versões, dependendo da classe (alternativa ou versão), através da função **composição por classe**.

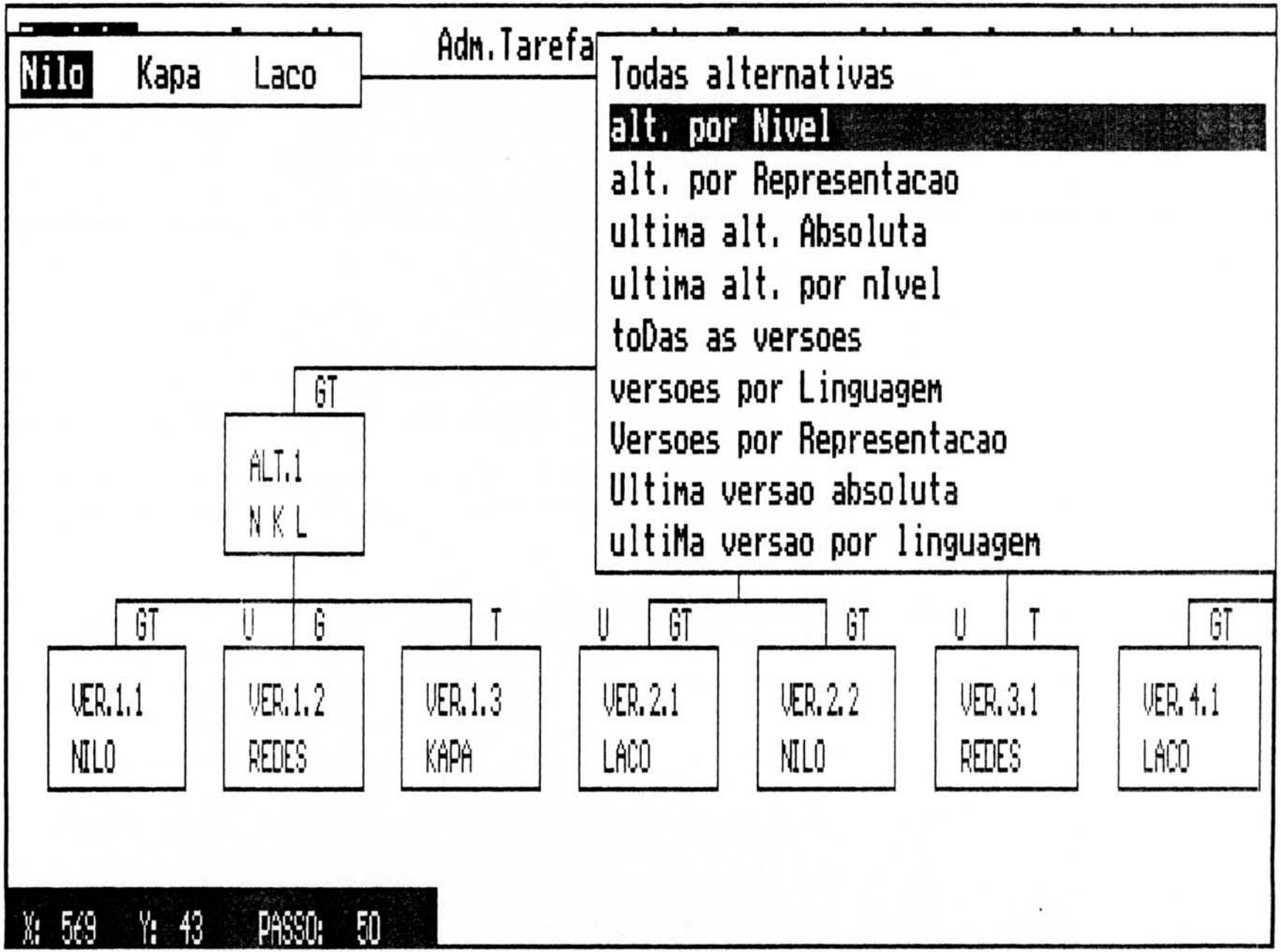


Figura 3.22: Funções de "poda" quando uma agência for apontada

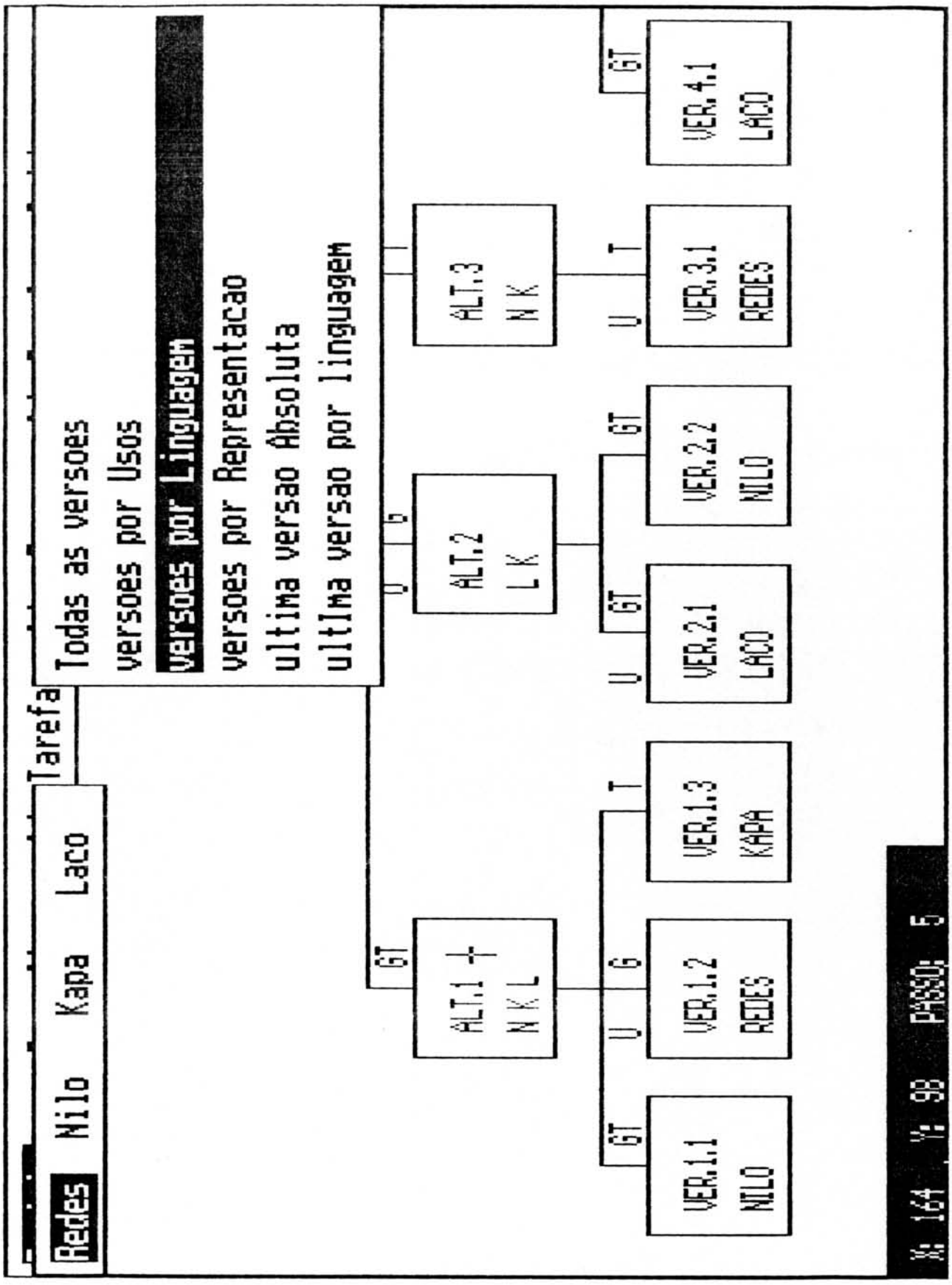


Figura 3.23: Funções de "poda" quando uma alternativa for apontada

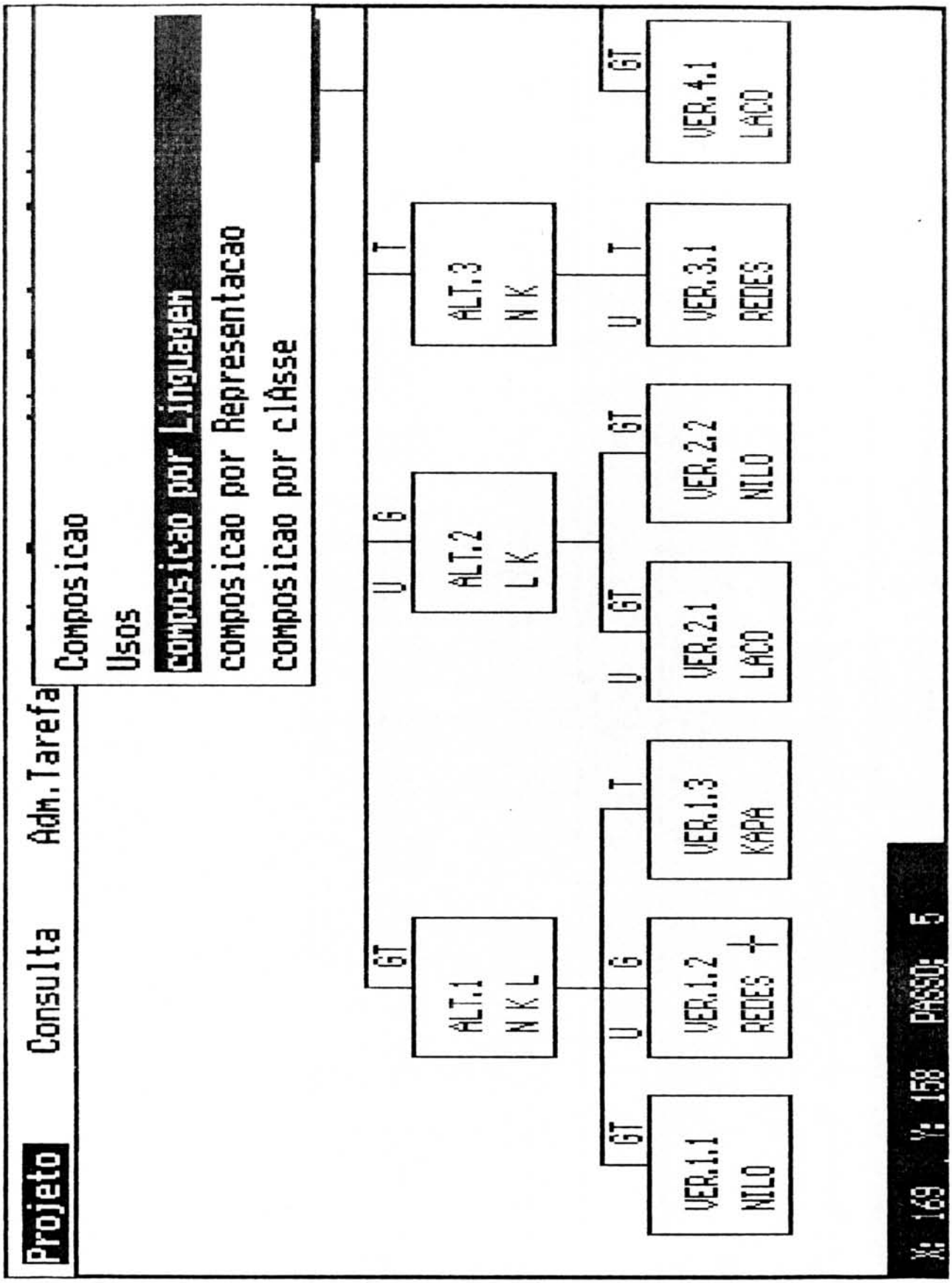


Figura 3.24: Funções de "poda" quando uma versão for apontada

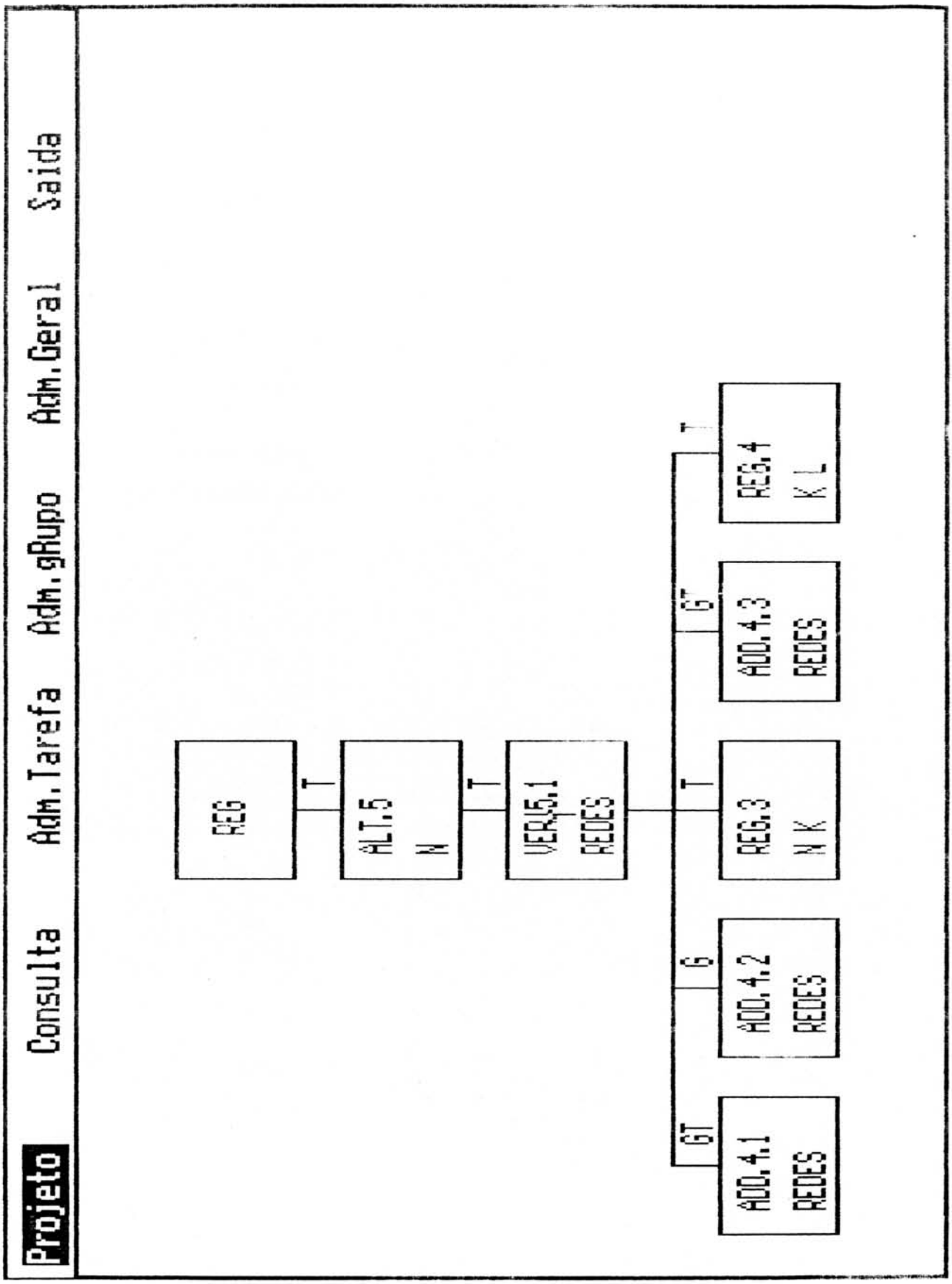


Figura 3.25: Arvore de composição, se uma versão composta for apontada

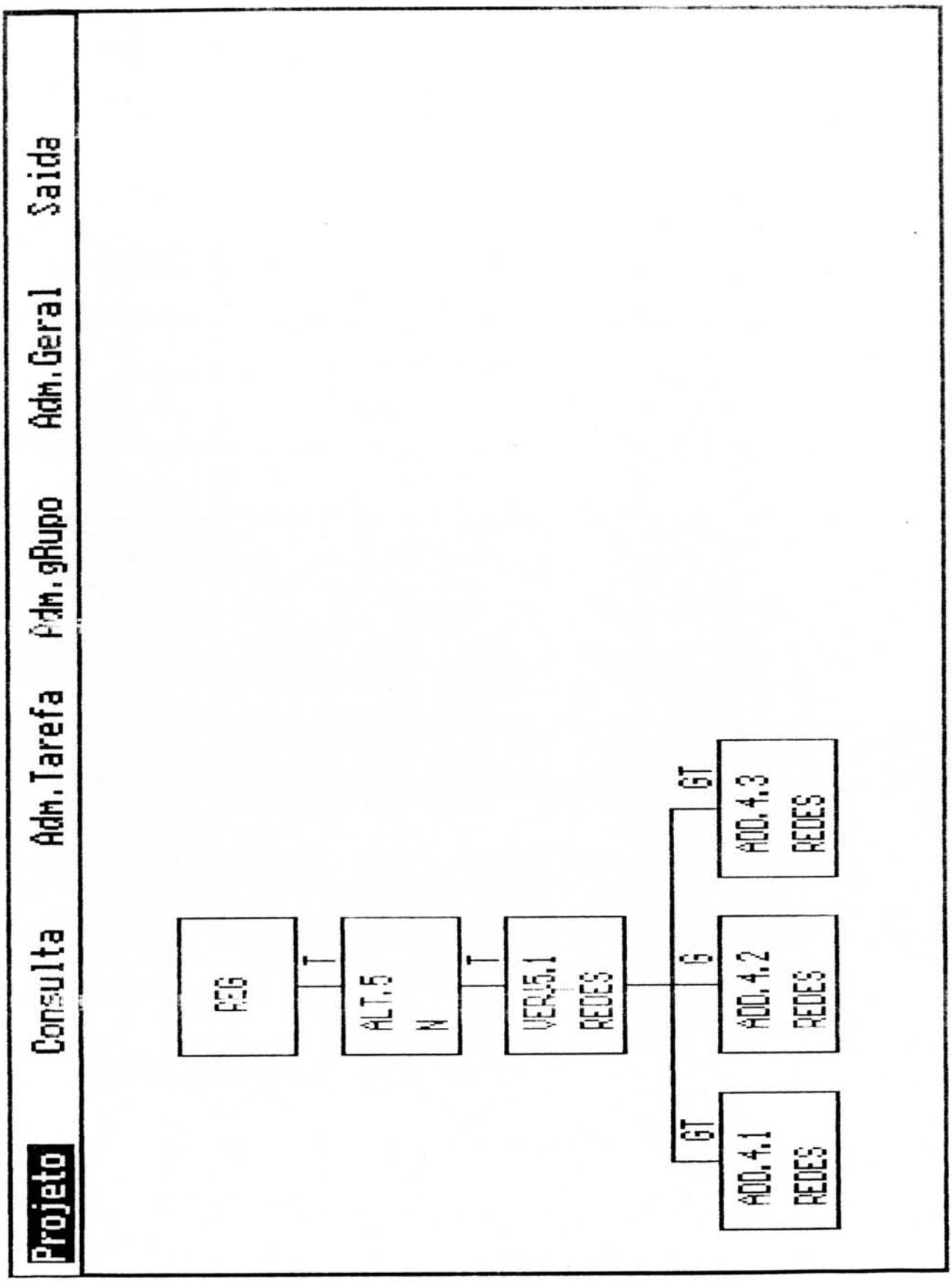


Figura 3.26: Arvore de usos, versões REDES onde o objeto apontado é usado

3.6 Comentários Finais

Comparativamente aos ambientes apresentados no capítulo 1, AMPLO possui: consistência dos dados de projeto manipulados por diferentes ferramentas, através de uma base de dados unificada, incluindo consistência entre diferentes representações de um mesmo sistema digital; uniformidade nas interfaces com o usuário apresentadas por diferentes ferramentas, incluindo facilidades de manipulação de janelas, caixas de diálogo, mensagens de erros; mecanismos de consulta e gerência da base de dados de projeto; controle de acesso ao sistema por parte de usuários, através de identificação e senha, e mecanismos de gerência de projeto.

Como os ambientes CWS e FACE, LAGO possui funções de gerência de projeto, objetos e usuários correntes, base de dados orientada a objetos e navegação na base de dados. Como o FACE, LAGO permite acesso a editores interativos.

ADAM é baseado em uma base de conhecimentos a nível de transferência entre registradores. Possui também um sistema especialista para testar regras de projeto de circuitos, facilidade que não está presente em AMPLO. Mas características como: produzir circuitos que possuam uma estrutura que possa ser simulada, interação uniforme com o usuário, projetos subdivididos em tarefas, projeto incremental e conceito de visão (ADAM) e contexto (AMPLO), são comuns a ambos os ambientes.

4 IMPLEMENTAÇÃO

4.1 Geração dos Cardápios de Funções de LAGO

A interface LAGO foi prototipada utilizando o Gerador de Interfaces orientadas a Menus (GIM) [BAG 89b].

E definida uma lista de menus com suas respectivas opções. Estes menus são dispostos hieraquicamente na forma de árvore, com funções de retorno ao menu de maior nível. O sistema GIM oferece ao programa de aplicação a possibilidade de redefinição das opções de um menu em tempo de execução, através da declaração de uma parte destas opções como variável. Cada opção do menu pode ser acessada posicionando o cursor diretamente ou através de um caracter que identifica uma opção. GIM permite inclusão de novas ferramentas e permite integração simples ao ambiente. GIM permite menor tempo dispendido na programação de uma interface [BAG 89b], permitindo também uma interface homogênea para todas as ferramentas.

O sistema GIM é dividido em três partes: compilação da descrição da interface, geração de estruturas de dados descritivas dos cardápios e geração das funções de manipulações dos cardápios. O sistema GIM tem como entrada uma descrição textual da interface. O processamento desta descrição é feito através de um compilador, sendo gerados um programa e um arquivo de "header", que devem ser incluídos no aplicativo. O apêndice A contém a descrição dos cardápios de LAGO na linguagem de definição de menus para o sistema GIM.

As opções de um cardápio podem ativar outros cardápios ou ativar uma função do aplicativo.

No caso de LAGO, as rotinas ativadas pela seleção de opções dos cardápios correspondem à implementação das funções constantes nas figuras 3.1 e de 3.22 à 3.24.

4.2 Conjunto de Rotinas de Gerência de Projeto e de Navegação na Base de Dados

A implementação de LAGO compreende um total de 6.237 linhas de código (Linguagem "C"), das quais 1.555 linhas fazem parte da interface gerada pelo GIM e 4.682 foram implementadas, neste trabalho, em "C". O software está organizado em diversos arquivos, como pode ser observado na figura abaixo:

Linhas	Arquivo	Função
2.381	LAGO.C	arquivo fonte principal
74	LAGO.H	protótipos das funções de "LAGO"
102	LAGODEFS.H	definições das constantes
75	LAGOTIPO.H	define estruturas e variáveis
3	LAGO.PRJ	arquivo de "projeto"
239	LAGO.DAT	descrição de "LAGO" para o "GIM"
25	BROWSER.DEF	define constantes do "Browser"
144	BROWSER.TIP	define var. e estrut. "Browser"
1.290	INTERFAC.C	interface gerada pelo "GIM"
265	INTERFAC.H	protótipos do "INTERFACE.C"
1.545	LAGOROT.C	rotinas auxiliares de "LAGO"
94	LAGOROT.H	protótipos de "LAGOROT.C"

Figura 4.1: Programas e arquivos de LAGO

LAGO corresponde, na verdade, a um conjunto de rotinas que implementam todas as funções ativáveis pelo usuário, à exceção das funções de administração de grupo (ver apêndice D, D.2.3 à D.2.12) e de tarefa (ver apêndice D, D.3.1 à D.3.9). Estas rotinas não foram implementadas uma vez que a interface com o BD existente não as suporta ainda (ver figura 4.2).

Administracao Geral	Banco de Dados
Cria usuario	CRIA_USUARIO(ent:ius,senha)
Remove usuario	REMOVE_USUARIO(ent:ius)
Lista	LISTA_USUARIOS(ent:parametro)
crIa Grupo	CRIA_GR(ent:gr,adm,lst_prj)
rEmove Grupo	REMOVE_GR(ent:gr)
Muda/administrador	ALTERA_ADM(ent:gr,adm)

Administracao Grupo	Banco de dados
inclusao Membros	INCLUI_US_GR(ent:gr,lst_us)
eXclusao membros	REMOVE_US_GR(ent:gr,lst_us)
Cria contexto	CRIA_CTX(ent:ctx,regra,lst)
Remove contexto	REMOVE_CTX(ent:ctx)
Inclui objeto contexto ...	INCLUI_OBJ(ent:ctx,obj)
Exclui objeto contexto ...	REMOVE_OBJ(ent:ctx,obj)
aSsocia contexto	AUTORIZA(ent:ctx,gr)
Desassocia contexto	DESAUTORIZA(ent:ctx,gr)
Libera objeto	CONSOLIDA_OBJ(ent:idp,obj)
iNicia projeto	INICIA_TP(ent:idp,idu)
Finaliza projeto	FIM_TP(ent:idp)
cancela Projeto	CANCELA_TP(ent:idp)

Administracao Tarefa	Banco de dados
Inicia tarefa	INCLUI_TU(ent:idp,idu)
iNterrompe tarefa	INTERROMPE_TU(ent:idp,idu)
Continua tarefa	CONTINUA_TU(ent:idp,idu)
Finaliza tarefa	FIM_TU(ent:idp,idu)
Libera objeto	CONSOLIDA_OBJ(ent:idp,obj)
Empresta objeto	EMPRESTA_OBJ(ent:idp,idu_o,idu_d,obj)
Transfere objeto	ESTABILIZA_OBJ(ent:idp,idu,obj)
Devolve objeto	DEVOLVE_OBJ(ent:idp,idu_o,idu_d,obj)
Busca objeto	BUSCA_OBJ(ent:idp,idu_o,idu_d,obj)

lst_us ..... lista de usuarios
gr ..... grupo
idp ..... transacao de projeto
idu ..... transacao de usuario
ctx ..... contexto
obj ..... objeto
idu_o ..... transacao de usuario origem
idu_d ..... transacao de usuario destino
ius ..... identificacao do usuario
adm ..... administrador de grupo
lst_prj ..... lista de projetistas do grupo

Figura 4.2: Equivalência entre as funções de LAGO e as do banco de dados

As rotinas implementadas (ver figuras 4.3 e 4.4) manipulam arquivos de cadastro de usuários e grupos ("user.dat" e "grupos.dat") e arquivo auxiliares para a navegação. Tais arquivos encontram-se descritos no apêndice C. Particularmente, as estruturas de dados para a navegação são montadas em memória a partir de arquivos que contêm objetos de teste e não a base de dados real do AMPLO face à sua indisponibilidade no momento de implementação.

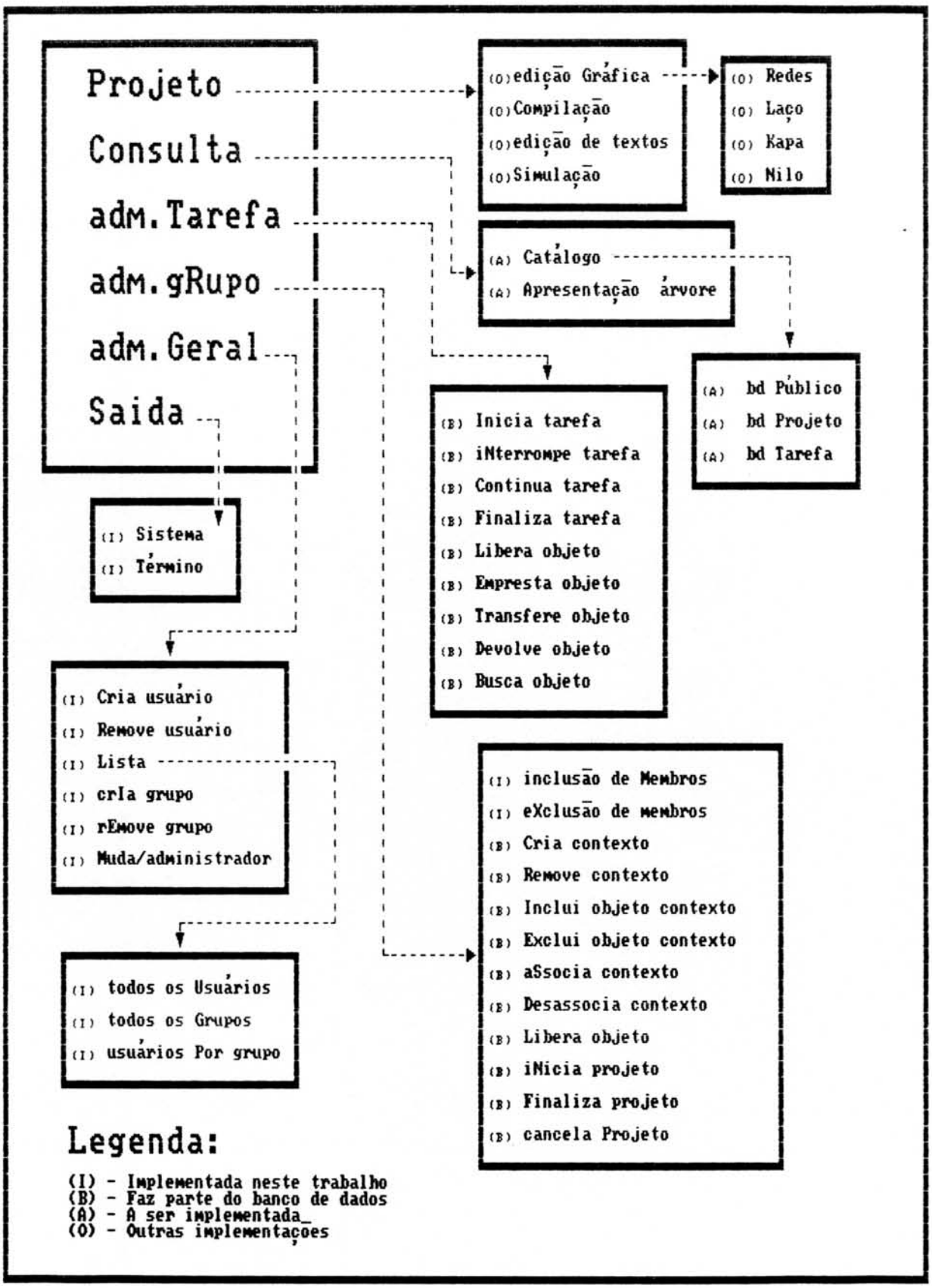


Figura 4.3: Situação da implementação das Rotinas de Gerência

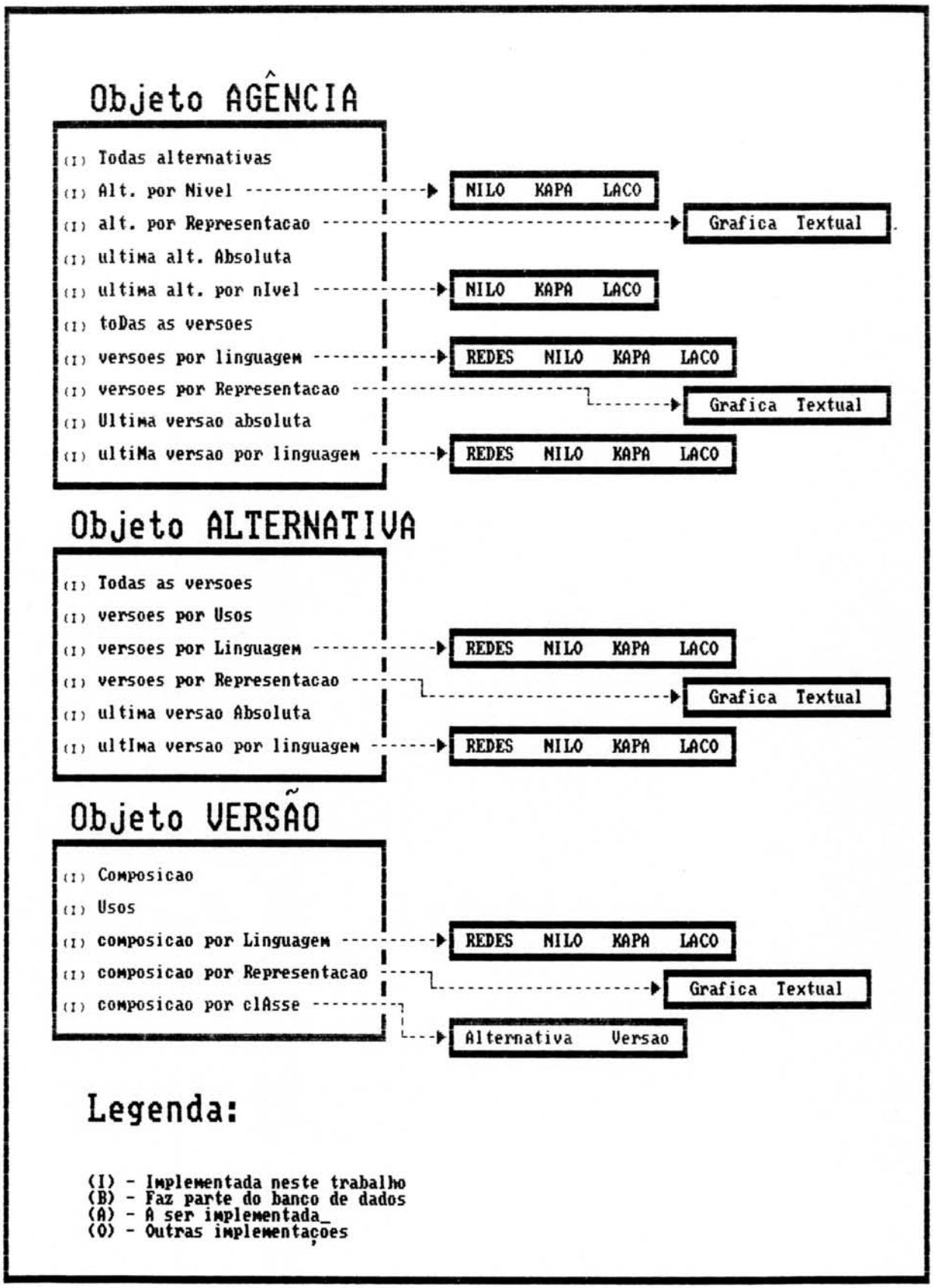


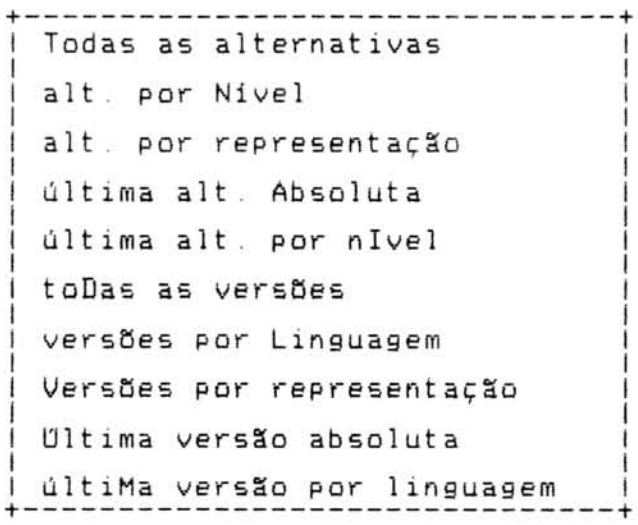
Figura 4.4: Situação da implementação das Rotinas de Navegação

4.3 Aspectos Principais do Funcionamento do Sistema

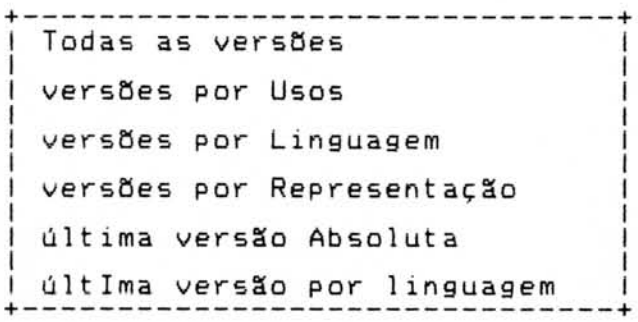
4.3.1 Detalhamento da Implementação das Rotinas Gráficas de Navegação de LAGO

a) Programa Principal:

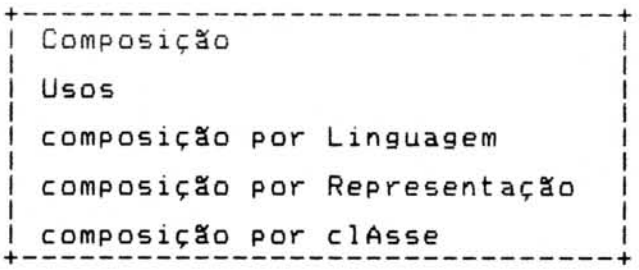
- inicializa as variáveis;
- monta a tela inicial (ver figura 3.2);
- solicita a identificação do usuário;
- testa se o usuário está cadastrado; se não estiver, é exibida a seguinte mensagem: "Usuário não cadastrado" (ver figura 3.20);
- solicita a senha do usuário (ver figura 3.3);
- testa se a senha confere com o usuário; se não conferir, é exibida a seguinte mensagem: "Senha inválida";
- exibe sistema de menus (ver figura 3.5);
- carrega a estrutura com as informações das agências;
- calcula a posição dos retângulos que identificam as agências;
- visualiza o diretório com todas as agências, permitindo "scroll" (ver figura 3.5);
- permite apontamento de uma agência em especial;
- exibe a árvore de composição da agência apontada (ver figura 3.15) e carrega todas as informações sobre as alternativas e versões nas estruturas correspondentes;
- permite apontamento na árvore de composição, que contém: agência, alternativas e versões;
- o usuário pode apontar: agência, alternativa ou versão composta ("REDES");
  - se uma **agência** for apontada, é exibido um cardápio vertical, com as seguintes opções: (ver figura 3.22);



- se uma **alternativa** for apontada, é exibido um cardápio vertical, com as seguintes opções (ver figura 3.23);



- se uma **versão REDES** for apontada, é exibido um cardápio vertical, com as seguintes opções (ver figura 3.24);



- ERRO: se nenhum dos objetos: agência, alternativa ou versão for apontado, é exibida a seguinte mensagem de erro: "Objeto MAL apontado" ou "OBJETO apontado não possui VERSÃO COMPOSTA";

b) Funções ativadas pelo programa principal:

Função: TODAS AS ALTERNATIVAS (figura 3.22)
Rotina: R_ag_todas_alternativas();

- desativa todas as versões (na estrutura que contém os dados correspondente à versão);
- ativa todas as alternativas (na estrutura que contém os dados correspondente à alternativa);
- calcula a posição dos retângulos que representam as alternativas e a agência correspondente;
- visualiza a árvore de composição da agência com todas as alternativas;
- permite apontamento da agência ou qualquer alternativa da árvore de composição exibida na tela;

Função: ALTERNATIVAS POR NIVEL (figura 3.22)
Rotinas: R_alt_nível_nilo();
R_alt_nível_kapa();
R_alt_nível_laco();

- desativa todas as versões;
- desativa todas as alternativas;
- ativa todas as alternativas que possuem o nível selecionado ('N': NILO, 'K': KAPA ou 'L': LAÇO);
- calcula a posição dos retângulos que representam as alternativas e o retângulo que representa a agência correspondente ao objeto apontado;
- visualiza a árvore de composição da agência com todas as alternativas que possuem o nível selecionado;
- permite apontamento da agência ou qualquer alternativa da árvore de composição exibida na tela;

Função: ALTERNATIVAS POR REPRESENTAÇÃO (figura 3.22)
Rotinas: R_alt_representação_gráfica();
R_alt_representação_textual();

- desativa todas as alternativas;
- desativa todas as versões;
- ativa todas as versões que contém o tipo de representação ('G': GRAFICA ou 'T': TEXTUAL);
- calcula a posição dos retângulos que representam as alternativas e o retângulo que representa a agência correspondente a alternativa selecionada;
- visualiza a árvore de composição da agência com todas as alternativas que possuem a representação selecionada;
- permite apontamento da agência ou qualquer alternativa da árvore de composição exibida na tela;

Função: ÚLTIMA ALTERNATIVA ABSOLUTA (figura 3.22)
Rotina: R_ag_última_alternativa_absoluta();

- desativa todas as alternativas;
- desativa todas as versões;
- pesquisa todas as alternativas e verifica qual é a última alternativa absoluta;
- ativa todas as versões correspondentes a última alternativa absoluta;
- recalcula a posição de todos os objetos;
- visualiza a árvore de composição, que contém os objetos: agência, última alternativa absoluta e versões;
- permite apontamento dos objetos: agência, alternativa ou versões, que estão exibidos na tela;

Função: ÚLTIMA ALTERNATIVA POR NIVEL (figura 3.22)
Rotinas: R_alt_ult_nível_nilo();
R_alt_ult_nível_kapa();
R_alt_ult_nivel_laco();

- desativa todas as alternativas;
- desativa todas as versões;
- pesquisa todas as alternativas e verifica a última alternativa correspondente ao nível selecionado ('N': NILO, 'K': KAPA ou 'L': LAÇO);

- calcula a posição dos retângulos que representam as alternativas e o retângulo que representa a agência correspondente à alternativa selecionada;
- visualiza a árvore de composição da agência com as últimas alternativas por nível;
- permite apontamento da agência ou qualquer alternativa da árvore de composição exibida na tela;

Função: TODAS AS VERSOES (figura 3.22)
Rotina: R_ag_todas_versões

- calcula a posição dos retângulos que representam as versões;
- calcula a posição dos retângulos que representam as alternativas;
- calcula a posição do retângulo que representa a agência;
- visualiza a árvore de composição, que contém os objetos: agência, todas as alternativas e todas as versões;
- permite apontamento da agência, das alternativas ou das versões;

Função: VERSOES POR LINGUAGEM (figura 3.22)
Rotinas: R_ver_linguagem_redes();
R_ver_linguagem_nilo();
R_ver_linguagem_kapa();
R_ver_linguagem_laço();

- desativa todas as alternativas;
- desativa todas as versões;
- pesquisa todas as versões e ativa as que correspondem ao nível escolhido, ativando também, a alternativa correspondente;
- calcula a posição dos retângulos que representam as versões;
- calcula a posição dos retângulos que representam as alternativas;
- calcula a posição do retângulo que representa a agência;

- visualiza a árvore de composição, que contém os objetos: agência, alternativas e versões correspondentes ao nível escolhido;
- permite apontamento da agência, das alternativas ou das versões;

Função: VERSOES POR REPRESENTAÇAO (figura 3.22)
Rotinas: R_ver_representacao_grafica();
R_ver_representacao_textual();

- desativa todas as alternativas;
- desativa todas as versões;
- pesquisa todas as versões e ativa as que contém o tipo de representação escolhida ('G': Gráfica ou 'T': Textual), ativando também, a alternativa correspondente;
- calcula a posição dos retângulos que representam: versões, alternativas e o retângulo que representa a agência correspondente a alternativa escolhida;
- visualiza a árvore de composição, que contém os objetos: agência, alternativas e versões;
- permite apontamento da agência, das alternativas ou das versões;

Função: ÚLTIMA VERSAO ABSOLUTA (figura 3.22)
Rotina: R_ag_ultima_versão_absoluta();

- desativa todas as alternativas;
- desativa todas as versões;
- pesquisa todas as versões, verificando e ativando a última versão absoluta;
- ativa a alternativa correspondente;
- recalcula a posição dos objetos;
- visualiza a árvore de composição, que contém os objetos: agência, última versão absoluta e alternativa;
- permite apontamento da agência, da alternativa ou da versão;

Função: ÚLTIMA VERSÃO POR LINGUAGEM (figura 3.22)
Rotinas: R_ver_ult_linguagem_redes();
R_ver_ult_linguagem_nilo();
R_ver_ult_linguagem_kapa();
R_ver_ult_linguagem_laco();

- desativa todas as alternativas;
- desativa todas as versões;
- pesquisa todas as versões, verificando e ativando a última versão da linguagem selecionada ('R': REDES, 'N': NILO, 'K': KAPA ou 'L': LAÇO);
- calcula a posição dos retângulos que representam as alternativas, versões e agência;
- visualiza a árvore de composição, que contém os objetos: agência, alternativas e versões, que possuem a última versão por linguagem escolhida;
- permite apontamento da agência, das alternativas ou versões;

Função: VERSOES POR USOS (figura 3.23)
Rotina: R_alt_versões_por_usos();

- desativa todas as alternativas;
- desativa todas as versões;
- desativa todas as ocorrências;
- ativa a versão apontada e a alternativa correspondente;
- pesquisa todas as ocorrências e ativa as que possuem versão composta na linguagem REDES;
- calcula a posição dos retângulos que representam as ocorrências;
- calcula a posição da versão, da alternativa e da agência correspondente;
- visualiza a árvore de composição, que contém os objetos: agência, alternativa, versão e ocorrências;
- permite apontamento dos objetos: agência, alternativa ou versão;

Função: COMPOSIÇÃO (figura 3.24)
Rotinas: R_ver_composição();

- desativa todas as alternativas;
- desativa todas as versões;
- desativa todas as ocorrências;
- ativa a versão apontada e a alternativa correspondente;
- pesquisa e ativa todas as ocorrências correspondentes a versão composta apontada;
- calcula a posição dos retângulos que representam as ocorrências;
- calcula a posição da versão, alternativa e agência;
- visualiza a árvore de composição, que contém os objetos: agência, alternativa, versão e ocorrências;
- permite apontamento da agência, alternativa ou versão;

Função: COMPOSIÇÃO POR LINGUAGEM (figura 3.24)
Rotinas: R_comp_linguagem_redes();
R_comp_linguagem_nilo();
R_comp_linguagem_kapa();
R_comp_linguagem_laço();

- desativa todas as alternativas;
- desativa todas as versões;
- desativa todas as ocorrências;
- ativa a versão apontada e a alternativa correspondente;
- pesquisa e ativa todas as ocorrências correspondentes a linguagem selecionada ('R': REDES, 'N': NILO, 'K': KAPA ou 'L': LAÇO);
- calcula a posição dos retângulos que representam as ocorrências;
- calcula a posição dos retângulos que representam a versão, alternativa e agência;
- visualiza a árvore de composição, que contém os objetos: agência, alternativa, versão e ocorrências dependendo da linguagem selecionada;
- permite apontamento da agência, alternativa ou versão;

Função: COMPOSIÇÃO POR REPRESENTAÇÃO (figura 3.24)
Rotinas: R_comp_representação_gráfica();
R_comp_representação_textual();

- desativa todas as alternativas;
- desativa todas as versões;
- desativa todas as ocorrências;
- ativa a versão apontada e a alternativa correspondente;
- pesquisa e ativa todas as ocorrências correspondentes a representação selecionada ('G': GRAFICA ou 'T': TEXTUAL);
- calcula a posição dos retângulos que representam as ocorrências;
- calcula a posição dos retângulos que representam a versão, alternativa e agência;
- visualiza a árvore de composição, que contém os objetos: agência, alternativa, versão e ocorrências dependendo da representação selecionada;
- permite apontamento da agência, alternativa ou versão;

Função: COMPOSIÇÃO POR CLASSE (figura 3.24)
Rotinas: R_comp_classe_alternativa();
R_comp_classe_versão();

- desativa todas as alternativas;
- desativa todas as versões;
- desativa todas as ocorrências;
- ativa a versão apontada e a alternativa correspondente;
- pesquisa e ativa todas as ocorrências correspondentes a representação selecionada ('G': GRAFICA ou 'T': TEXTUAL);
- calcula a posição dos retângulos que representam as ocorrências;
- calcula a posição dos retângulos que representam os objetos: versão, alternativa e agência correspondente;
- visualiza a árvore de composição, que contém os objetos: agência, alternativa, versão e ocorrências dependendo da representação selecionada;
- permite apontamento da agência, alternativa ou versão;

4.3.2 Detalhes da Visualização Gráfica dos Objetos

A seguir são dados detalhes sobre a exibição do diretório de agências. Os objetos na tela estão dispostos da seguinte maneira:

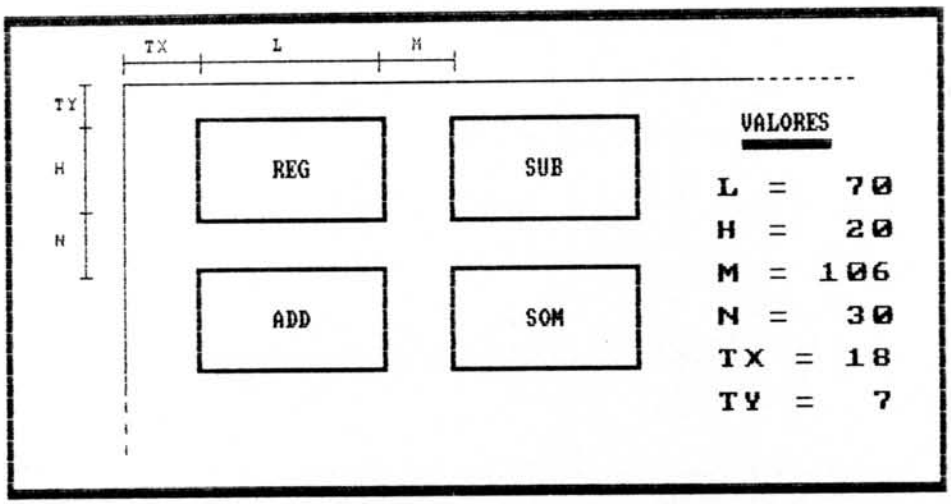


Figura 4.5: Posicionamento dos objetos na Representação Gráfica do Diretório de Agências

```
nro_colunas = sqrt(nro_agencias);
for (i = 0;i (= nro_agencias - 1;i++)
    {
     lin = i / nro_colunas;
     col = i % nro_colunas;
     AGE[i].x1 = TX + M * col;
     AGE[i].y1 = TY + N * lin;
     AGE[i].x2 = AGE[i].x1 + L;
     AGE[i].y2 = AGE[i].y1 + H;
    }
```

A seguir são dados detalhes sobre a exibição da árvore de composição da agências. O posicionamento dos objetos na tela é feito de baixo para cima, ou seja, do último nível ao primeiro nível, da seguinte maneira:

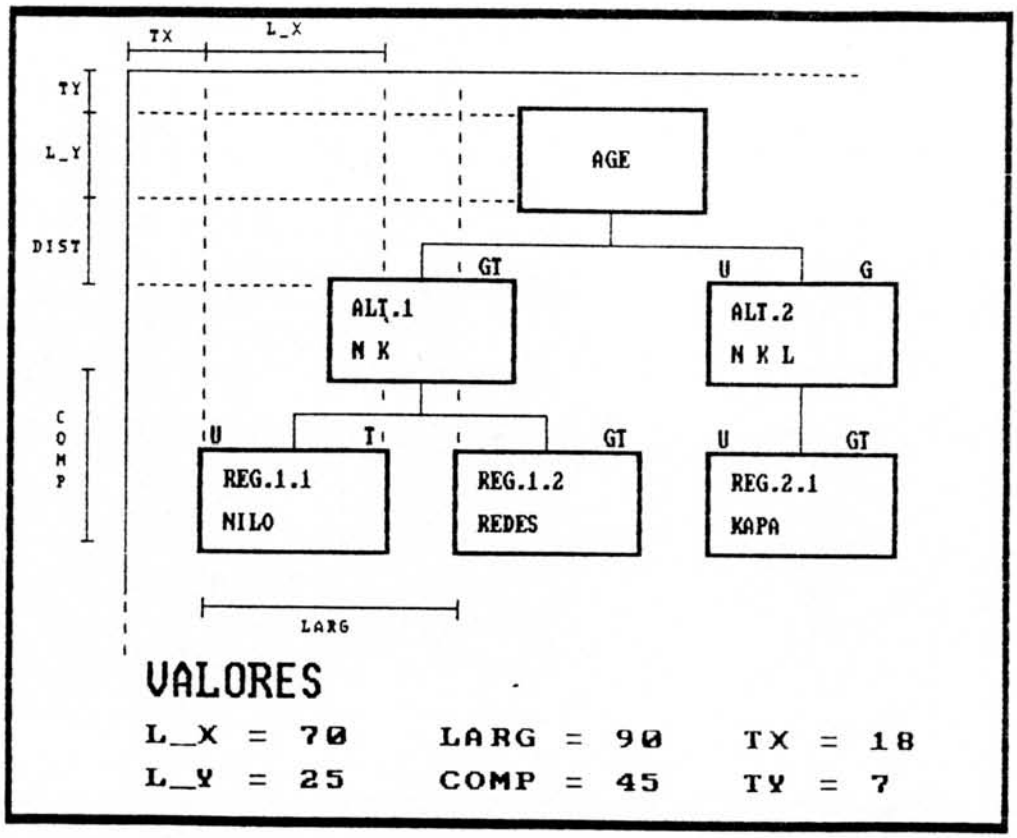


Figura 4.6: Posicionamento dos objetos na Arvore de Composição de uma Agência

O cálculo do posicionamento dos objetos é feito da seguinte maneira:

Cálculo das posições das VERSÕES:

```
dist = COMP - L_Y;
j = 0;
for (i = 0;i <= nro_versoes - 1;i++)
    {
     VER[i].x1 = LARG - L_X + LARG * j;
     VER[i].y1 = 2 * COMP + dist;
     VER[i].x2 = VER[i].x1 + L_X;
     VER[i].y2 = VER[i].y1 + L_Y;
     j++;
    }
```

Cálculo das posições das ALTERNATIVAS:

```
for (i = 0;i <= nro_alternativas - 1;i++)
    {
     VERSOES_ATIVAS(i,&ver_1,&ver_2);
     if (ver_1 != ver_2)
        {
         ALT[i].x1 = (VER[ver_2].x2 + VER[ver_1].x1) / 2 -
                     (L_X / 2);
         ALT[i].y1 = COMP + dist;
         ALT[i].x2 = ALT[i].x1 + L_X;
         ALT[i].y2 = ALT[i].y1 + L_Y;
        }
     else
        {
         ALT[i].x1 = VER[ver_1].x1;
         ALT[i].y1 = COMP + dist;
         ALT[i].x2 = ALT[i].x1 + L_X;
         ALT[i].y2 = ALT[i].y1 + L_Y;
        }
    }
```

Cálculo da posição da AGENCIA:

```
ALTERNATIVAS_ATIVAS(&alt_1,&alt_2);
if (alt_1 != alt_2)
   {
    AGE[AGENCIA].x1 = (ALT[alt_2].x1 + ALT[alt_1].x1 / 2;
    AGE[AGENCIA].y1 = dist;
    AGE[AGENCIA].x2 = AGE[AGENCIA].x1 + L_X;
    AGE[AGENCIA].y2 = AGE[AGENCIA].y1 + L_Y;
   }
else
   {
    AGE[AGENCIA].x1 = ALT[alt_1].x1;
    AGE[AGENCIA].y1 = dist;
    AGE[AGENCIA].x2 = AGE[AGENCIA].x1 + L_X;
    AGE[AGENCIA].y2 = AGE[AGENCIA].y1 + L_Y;
   }
```

5 CONCLUSÕES E AVALIAÇÕES

LAGO foi desenvolvida como a interface de alto nível de AMPLO. Abriga funções de ativação de ferramentas de projeto, funções de controle de acesso ao ambiente, funções de controle de objetos das bases de dados privativa, de projeto e pública e, também, funções de consulta a esses três níveis de base de dados. Estas funções de LAGO são completas e consistentes em relação às interfaces de ambientes de projeto de sistemas digitais existentes ou propostos.

A interface foi concebida segundo as modernas tendências de interação, com as funções de nível mais alto num cardápio horizontal superior, menus "pull down" e caixas de diálogo. Na época em que foi iniciado o trabalho, as ferramentas de apoio disponíveis em AMPLO (PG [OLA 88b], IS [OLA OLA 87a], IE [OLA 87b, OLA 87c] e PIU [OLA 89]) não ofereciam a flexibilidade necessária para esse tipo de projeto. Após o estudo destas ferramentas e de outras disponíveis na instituição, optou-se por desenvolver a primeira versão de LAGO utilizando o software GIM, conforme mencionado no capítulo 4, já que este gera uma interface idêntica àquela especificada para LAGO.

Neste ponto, esta primeira versão perde um pouco a característica de integração com as demais ferramentas de projeto do AMPLO, cujas interfaces tiveram seu projeto feito há já alguns anos.

Uma característica muito usada em ambientes interativos é a utilização de ícones para representar funções e objetos. No caso de LAGO, funções são representadas por textos, por imposição do software usado para produzir os cardápios; já objetos, no caso do "browser", são exibidos como ícones simples (retângulos com texto associado).

O ambiente, em si, ainda não apresenta a integração total das ferramentas. As funções de gerência de projeto (que manipulam transações de projeto e de usuários, contextos, grupos e transferência e empréstimo de objetos) que fazem parte da interface com o banco de dados, ainda não estão implementadas. Assim, muitas das funções em LAGO não puderam ser levadas a termo. Entretanto, todas as funções a nível de interface estão disponíveis aguardando apenas a ligação com as rotinas correspondentes da interface com o banco de dados.

A integração das ferramentas de projeto, editores, compiladores e simuladores ao banco de dados tem sido realizada de forma lenta. O ambiente, em si, ainda não está convenientemente integrado, muito mais por questões de infraestrutura laboratorial do que técnicas. Assim, a primeira versão de LAGO, para testar as funções de navegação, restringiu-se a prototipar agências, alternativas e versões em estruturas de dados internas e, a partir daí, permitir os diversos tipos de consulta descritos. As rotinas mais complexas do trabalho foram, portanto, implementadas, como por exemplo as rotinas de navegação, onde todos os objetos tem que ser calculados (posição dos retângulos, linhas horizontais e verticais e o texto que identifica as agências, alternativas, versões e ocorrências) e visualizados. Poucos detalhes ficaram para a segunda versão. Por exemplo, na função "Apresentação" (árvore ou lista) optou-se por implementar o "browser" com apresentação em árvore, que é mais complexa, deixando a apresentação em lista (mais simples) para futura extensão.

A primeira extensão do presente trabalho e', portanto, colocar LAGO sobre a base de dados quando se dispuser da implementação dos conceitos de gerência de projeto. A segunda extensão é reestruturar LAGO sobre as ferramentas gráficas de AMPLO (PIU, PG, IS e IE) integrando de fato o ambiente do ponto de vista da homogeneidade da interface

com o usuário.

Resta, ainda, um último nível de integração, com o ambiente de simulação, que está sendo desenvolvido em separado por sua inerente complexidade. Neste ambiente, são necessários os mesmos mecanismos de navegação sobre a base de dados, para permitir consultas e construção de novos objetos (modelos de simulação). A estrutura navegacional deve, portanto, ser integrada. Além disto, o ambiente de simulação também disporá de uma interface de alto nível com o usuário, que dará acesso a várias funções específicas de gerência de projeto, e que deverá manter o estilo de interação adotado em LAGO.

A interface de alto nível LAGO não está completa, mas todo o ambiente foi especificado, ficando a tarefa de integração com o banco de dados e as ferramentas de AMPLO para uma segunda versão.

ANEXO A: Listagem da Especificação de LAGO para o GIM

```
( MENUS PROJETO AMPLO )

GERA C,GRAFICO (MOUSE LIGHTCYAN,POSICIONA,REFAZ);

CARACTERES

Car1 DEFAULTFONT;

CORES (LETRA,FUNDO,BORDA,SOMBRA,CAR ID,REVERSO,OFF)
Cor1    WHITE,BLUE,WHITE,WHITE,CYAN,LIGHTCYAN,CYAN;
Cor2    WHITE,BLUE,WHITE,WHITE,CYAN,LIGHTCYAN,CYAN;
Chelp   BLUE,WHITE,BLUE,BLACK,BLACK,BLACK,BLUE;

MENUS
Menu1
POSICAO 0,0,639,15;
COR      Cor1;
SENTIDO  HORIZONTAL;
'Projeto',          P, Mprojeto,R_projeto;
'Consulta',         C, Mconsulta;
'Adm.Tarefa',       T, Madm_tarefa;
'Adm.gRupo',        R, Madm_grupo;
'Adm.Geral',        G, Madm_geral;
'Saida',            S, Msaida;

Madm_geral
COR Cor2;
SENTIDO VERTICAL;
'Cria usuario',         C, NULO, R_cria_usuario;
'Remove usuario',       R, NULO, R_remove_usuario;
'Lista',                L, Mlista_dados, NULO;
'crIa grupo',           I, NULO, R_cria_grupo;
'rEmove grupo',         E, NULO, R_remove_grupo;
'Muda/administrador', M, NULO, R_mudanca_administrador;
```

```
Mlista_dados
COR Cor2;
SENTIDO VERTICAL;
'todos os Usuarios',      U, NULO, R_lista_todos;
'todos os Grupos',        G, NULO, R_lista_grupos;
'usuarios Por grupo',     P, NULO, R_lista_por_grupo;

Madm_grupo
COR Cor2;
SENTIDO VERTICAL;
'inclusao de Membros',
          M, NULO, R_inclusao_membros;
'eXclusao de membros',
          X, NULO, R_exclusao_membros;
'Cria contexto',
          C, NULO, R_cria_contexto,OFF;
'Remove contexto',
          R, NULO, R_remove_contexto,OFF;
'Inclui objeto contexto',
          I, NULO, R_inclui_objeto_contexto,OFF;
'Exclui objeto contexto',
          E, NULO, R_exclui_objeto_contexto,OFF;
'aSsocia contexto',
          S, NULO, R_associa_contexto_grupo,OFF;
'Desassocia contexto',
          D, NULO, R_desassocia_contexto_grupo,OFF;
'Libera objeto',
          L, NULO, R_libera_objeto,OFF;
'iNicia projeto',
          N, NULO, R_inicia_projeto,OFF;
'Finaliza projeto',
          F, NULO, R_finaliza_projeto,OFF;
'cancela Projeto',
          P, NULO, R_cancela_projeto,OFF;
```

```
Madm_tarefa
COR Cor2;
SENTIDO VERTICAL;
'Inicia tarefa',          I, NULO, R_inicia_tarefa,OFF;
'iNterrompe tarefa',      N, NULO, R_interrompe_tarefa,OFF;
'Continua tarefa',        C, NULO, R_continua_tarefa,OFF;
'Finaliza tarefa',        F, NULO, R_finaliza_tarefa,OFF;
'Libera objeto',          L, NULO, R_libera_objeto,OFF;
'Empresta objeto',        E, NULO, R_empresta_objeto,OFF;
'Transfere objeto',       T, NULO, R_transfere_objeto,OFF;
'Devolve objeto',         D, NULO, R_devolve_objeto,OFF;
'Busca objeto',           B, NULO, R_busca_objeto;

Mprojeto
COR Cor2;
SENTIDO VERTICAL;
'edicao Grafica',     G, Meditores_graficos, NULO;
'Compilacao',         C, NULO, R_compiladores;
'edicao de Textos',   T, NULO, R_editor_textos;
'Simulacao',          S, NULO, R_simulacao;

Mconsulta
COR Cor2;
SENTIDO VERTICAL;
'Catalogo      ',       C, Mcatalogo, NULO;
'Apresentacao  ',       A, M_apresentacao, NULO;

M_apresentacao
POSICAO 1,16,200,31;
COR      Cor1;
SENTIDO  HORIZONTAL;
'Arvore',       A, NULO, R_apresentacao_arvore;
'Lista',        L, NULO, R_apresentacao_lista;
```

```
Meditores_graficos
COR Cor2;
SENTIDO VERTICAL;
'Redes',   R, NULO, R_redes;
'Laco',    L, NULO, R_laco,OFF;
'Kapa',    K, NULO, R_kapa,OFF;
'Nilo',    N, NULO, R_nilo;

Mcatalogo
COR Cor2;
SENTIDO VERTICAL;
'bd Publico',       P, NULO, R_bd_publico;
'Contexto',         C, NULO, R_bd_projeto;
'bd Tarefa',        T, NULO, R_projetista;


Msaida
COR Cor2;
SENTIDO VERTICAL;
'Sistema',       1, NULO, R_sistema;
'Termino',       1, NULO, R_termino;

Menu2
POSICAO 300,16,639,130;
COR      Cor1;
SENTIDO  VERTICAL;
'Todas alternativas',
         T, NULO,R_ag_todas_alternativas;
'alt. por Nivel ',
         N, M_alt_nivel, NULO;
'alt. por Representacao ',
         R, M_alt_representacao, NULO;
'ultima alt. Absoluta',
         A, NULO,R_ag_ultima_alternativa_absoluta;
'ultima alt. por nIvel ',
         I, M_alt_ult_nivel, NULO;
'toDas as versoes',
         D, NULO,R_ag_todas_versoes;
'versoes por Linguagem ',
```

```
'Versoes por Representacao ',
        V, M_ver_representacao, NULO;
'Ultima versao absoluta',
        U, NULO,R_ag_ultima_versao_absoluta;
'ultiMa versao por linguagem ',
        M, M_ver_ult_linguagem, NULO;

M_alt_nivel
POSICAO 1,16,180,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Nilo',    N, NULO, R_alt_nivel_nilo;
'Kapa',    K, NULO, R_alt_nivel_kapa;
'Laco',    L, NULO, R_alt_nivel_laco;

M_alt_representacao
POSICAO 1,16,200,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Grafica',    G, NULO, R_alt_representacao_grafica;
'Textual',    T, NULO, R_alt_representacao_textual;

M_alt_ult_nivel
POSICAO 1,16,180,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Nilo',    N, NULO, R_alt_ult_nivel_nilo;
'Kapa',    K, NULO, R_alt_ult_nivel_kapa;
'Laco',    L, NULO, R_alt_ult_nivel_laco;

M_ver_linguagem
POSICAO 1,16,250,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Redes',   R, NULO, R_ver_linguagem_redes;
'Nilo',    N, NULO, R_ver_linguagem_nilo;
'Kapa',    K, NULO, R_ver_linguagem_kapa;
'Laco',    L, NULO, R_ver_linguagem_laco;
```

```
M_ver_representacao
POSICAO 1,16,200,31;
COR      Cor1;
SENTIDO  HORIZONTAL;
'Grafica',    G, NULO, R_ver_representacao_grafica;
'Textual',    T, NULO, R_ver_representacao_textual;

M_ver_ult_linguagem
POSICAO 1,16,250,31;
COR      Cor1;
SENTIDO  HORIZONTAL;
'Redes',    R, NULO, R_ver_ult_linguagem_redes;
'Nilo',     N, NULO, R_ver_ult_linguagem_nilo;
'Kapa',     K, NULO, R_ver_ult_linguagem_kapa;
'Laco',     L, NULO, R_ver_ult_linguagem_laco;

Menu3
POSICAO 300,16,639,89;
COR      Cor1;
SENTIDO  VERTICAL;
'Todas as versoes',
         T, NULO,R_alt_todas_versoes;
'versoes por Usos',
         U, NULO,R_alt_versoes_por_usos;
'versoes por Linguagem ',
         L, M_ver_linguagem, NULO;
'versoes por Representacao ',
         R, M_ver_representacao, NULO;
'ultima versao Absoluta',
         A, NULO,R_alt_ultima_versao_absoluta;
'ultIma versao por linguagem ',
         I, M_ver_ult_linguagem, NULO;

Menu4
POSICAO 300,16,639,75;
COR      Cor1;
SENTIDO  VERTICAL;

'Composicao',
```

```
'Usos',
        U, NULO,R_ver_usos;
'composicao por Linguagem ',
        L, M_comp_linguagem, NULO;
'composicao por Representacao ',
        R, M_comp_representacao, NULO;
'composicao por clAsse ',
        A, M_comp_classe, NULO;

M_comp_linguagem
POSICAO 16,1,200,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Redes',   R, NULO, R_comp_linguagem_redes;
'Nilo',    N, NULO, R_comp_linguagem_nilo;
'Kapa',    K, NULO, R_comp_linguagem_kapa;
'Laco',    L, NULO, R_comp_linguagem_laco;

M_comp_representacao
POSICAO 16,1,200,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Grafica',   G, NULO, R_comp_representacao_grafica;
'Textual',   T, NULO, R_comp_representacao_textual;

M_comp_classe
POSICAO 16,1,200,31;
COR     Cor1;
SENTIDO  HORIZONTAL;
'Alternativa',   A, NULO, R_comp_classe_alternativa;
'Versao',        V, NULO, R_comp_classe_versao;

Help
COR Chelp;
VERTICAL = 20;
HORIZONTAL = 60;
```

```
TECLAS
F1        HELP ('LAGO.HLP');
F10       Menu1;
ALT_P     Menu1,Mprojeto;
ALT_C     Menu1,Mconsulta;
ALT_T     Menu1,Madm_tarefa;
ALT_R     Menu1,Madm_grupo;
ALT_G     Menu1,Madm_geral;
ALT_S     Menu1,Msaida;
```

ANEXO B: Compilação do Software

As rotinas que implementam LAGO foram todas escritas em linguagem "C", usando o compilador "TURBO C 2.0".

LAGO é composto dos seguintes arquivos:

LAGO.C	fonte em linguagem do "Turbo C";
LAGO.H	protótipos das funções de "LAGO";
LAGODEFS.H	definições das constantes;
LAGOTIPO.H	definições das estruturas e variáveis;
LAGO.PRJ	arquivo de "projeto";
LAGO.DAT	arquivo com a descrição de "LAGO" para o gerenciador de interface "GIM";
BROWSER.DEF ...	definições das constantes do "Browser";
BROWSER.TIP ...	definições das variáveis globais e estruturas do "Browser";
INTERFAC.C	arquivo gerado pelo "GIM" para a interface de "LAGO";
INTERFAC.H	protótipos das funções da "INTERFACE";
LAGOROT.C	rotinas auxiliares de "LAGO";
LAGOROT.H	protótipos de "LAGOROT.C";
USER.DAT	arquivo de dados dos usuários;
GRUPOS.DAT	arquivo de dados dos grupos;
*.BGI	arquivos do "Turbo C" para tipos diferentes de placas de vídeo;
* CHR	arquivos do "Turbo C" para diferentes tipos de caracteres;
INIMOUSE.COM ..	arquivo do "Turbo C", para acessar o "mouse";
LAGO.EXE	arquivo executável;

Para compilar "LAGO", é necessário, em primeiro lugar, compilar "INTERFAC.C", gerando o arquivo objeto "INTERFAC.OBJ", e, logo após, compilar o arquivo "LAGOROT.C", gerando o arquivo "LAGOROT.OBJ". Deve-se ter um arquivo de "PROJECT", da seguinte maneira:

drive:\path\lago.c
drive:\path\interfac.obj
drive:\path\lagorot.obj

Carrega-se o arquivo "LAGO.C" no editor do "Turbo C", e constrói-se o arquivo "LAGO.EXE".

ANEXO C: Estruturas de Dados

As funções de LAGO fazem uso de diversas informações, armazenadas em arquivos. O arquivo "user.dat" contém a informação de todos os usuários com a seguinte estrutura:

```
struct  {
          char usuario[13];
          char senha[13];
          int grupo[10];
          int projetista;
          int pertence[10];
        } user;
```

Os nomes dos grupos criados pelo administrador geral constam no arquivo "grupos.dat", que contém a informação "nome_do_grupo".

Para realizar consultas à base de dados são necessárias informações sobre agências, alternativas, versões e ocorrências. Estas informações são armazenadas, durante a execução das funções de navegação, em estruturas auxiliares.

Para consultar as agências, temos:

```
struct agencia {
                 char age_nome[4];
                 int x1,y1;
                 int x2,y2;
                };
struct agencia *AGE;
```

Para consultar as alternativas, temos:

```
struct alternativa {
                    char alt_nome[8];
                    int x1;
                    int y1;
                    int x2;
                    int y2;
                    int alt_versoes;
                    int alt_ultima;
                    char alt_linguagem[6];
                    char alt_representação[3];
                    int alt_ativo;  /* 1 - ativo   */
                                    /* 0 - inativo */
                   };
struct alternativa *ALT;
```

Para consultar as versões, temos:

```
struct versao {
                char ver_nome[11];
                int x1;
                int y1;
                int x2;
                int y2;
                int ver_ultima;
                char ver_linguagem[6];
                char ver_representação[3];
                int ver_ativo;      /* 1 - ativo   */
                                    /* 0 - inativo */
               };
struct versao *VER;
```

Para consultar as ocorrências, temos:

```
struct ocorrencia  {
                     char oco_nome[11];
                     int x1;
                     int y1;
                     int x2;
                     int y2;
                     int oco_ultima;
                     char oco_linguagem[6];
                     char oco_representação[3];
                     int oco_ativo;   /* 1 - ativo    */
                                      /* 0 - inativo  */
                   };
struct ocorrencia *OCO;
```

ANEXO D: Descrição das Funções

D.1 Administração Geral (figura 3.6)

D.1.1 Cria Usuário

E solicitado o "nome_do_usuário". O usuário é criado ligado ao grupo; junto a ele é criada uma "senha" igual ao nome que, posteriormente, pode ser alterada (figura 3.6).

Entradas:

```
char *nome_do_grupo;
char *nome_do_usuário;
```

Rotina do BD:

```
CRIA_USUARIO(ent: ius, senha);
ius: identificação do usuário;
senha: senha do usuário;
```

D.1.2 Remove Usuário

E removido um usuário de AMPLO (figura 3.6).

Entrada:

```
char *nome_do_usuário;
```

Rotina do BD:

```
REMOVE_USUARIO(ent: ius);
ius: identificação do usuário;
```

D.1.3 Lista Usuários

E exibida uma janela, com três opções: "Lista todos os usuários" (todos os usuários do ambiente são listados),

"Todos os Grupos" (são listados os nomes de todos os grupos) e "Usuários por Grupo" (o usuário informa o "nome_do_grupo", e, então, são listados o nome do administrador do grupo e todos os usuários ligados ao grupo). Após a seleção de uma das opções, é exibida uma janela sobreposta com as informações. Para visualizar informações nesta janela usa-se: "PgUp" (página acima), "Pgdn" (página abaixo) e "ESC" (saída) (figuras 3.7,3.8,3.9 e 3.10).

Rotina do BD:

LISTA_USUARIOS(ent: parametro);
parametro: "todos" - todos os usuários são listados;
"gr" - todos os usuários do grupo são listados;

D.1.4 Cria Grupo

São solicitados "nome_do_grupo", "nome_do_administrador_do_grupo" e "nome_do_usuário" que farão parte do grupo, então é criado o grupo, com nome, administrador e projetistas ligados à ele (figura 3.6).

Entradas:

char *nome_do_grupo;
char *nome_do_administrador;
char *nome_do_usuário;

Rotina do BD:

CRIA_GR(ent: gr,adm,lst_prj);
gr: identificação do grupo;
adm: administrador do grupo;
lst_prj: lista dos usuários do grupo;

D.1.5 Remove Grupo

E solicitado o "nome_do_grupo" e é removido de AMPLO, permanecendo seus projetistas ligados a outros

grupos ou não (figura 3.6).

Entrada:

```
char *nome_do_grupo;
```

Rotina do BD:

```
REMOVE_GR(ent: gr);
gr: identificação do grupo;
```

D.1.6 Muda Administrador Grupo

São solicitados "nome_do_grupo" e o "nome_do_novo_administrador"; ocorre, então, uma troca na administração do grupo (figura 3.6).

Entradas:

```
char *nome_do_grupo;
char *nome_do_novo_administrador;
```

Rotina do BD:

```
ALTERA_ADM(ent: gr,adm);
gr: identificação do grupo;
adm: nome do novo administrador;
```

D.2 Administração de Grupo (figura 3.11)

D.2.1 Inclusão de Membros

São solicitados "nome_do_grupo" e o "nome_do_novo_usuário"; este usuário é ligado ao grupo (figura 3.11).

Entradas:

```
char *nome_do_grupo;
char *nome_do_usuário;
```

Rotina do BD:

```
INCLUI_US_GR(ent: gr,lst_us);
gr: identificação do grupo;
lst_us: lista dos usuários do grupo;
```

D.2.2 Exclusão de Membros

São solicitados "nome_do_grupo" e o "nome_do_usuário"; este usuário é removido do grupo (figura 3.11).

Entradas:

```
char *nome_do_grupo;
char *nome_do_usuário;
```

Rotina do BD:

```
REMOVE_US_GR(ent: gr,lst_us);
gr: identificação do grupo;
lst_us: lista dos usuários do grupo;
```

D.2.3 Cria Contexto

São solicitados "nome_do_contexto", a "regra" definidora do contexto ou a "lista" de objetos, e, então, é ativada a rotina correspondente da interface com o banco de

dados, para criar um contexto. E criada uma regra definidora do contexto, é especificada a lista de objetos incluídas no contexto (figura 3.11).

Entradas:

```
char *nome_do_contexto;
char *regra;
char *lista_de_objetos;
```

Rotina do BD:

```
CRIA_CTX(ent: ctx, regra, lst);
ctx: contexto
regra: regra definidora do contexto;
lst: lista de objetos definindo o contexto.
```

D.2.4 Remove Contexto

E solicitado o "nome_do_contexto", e, então, é ativada a rotina correspondente da interface com o banco de dados, para remover contextos, as referências ao contexto são removidas (figura 3.11).

Entrada:

```
char *nome_do_contexto;
```

Rotina do BD:

```
REMOVE_CTX(ent: ctx);
ctx: contexto.
```

D.2.5 Inclui Objeto Contexto

São solicitados "nome_do_contexto" e o "nome_do_objeto", e, então, é ativada a rotina correspondente da interface com o banco de dados, que inclui um objeto no contexto (figura 3.11).

Entradas:

```
char *nome_do_contexto;
char *nome_do_objeto;
```

Rotina do BD:

```
INCLUI_OBJ(ent: ctx, obj);
ctx: contexto;
obj: objeto a ser incluído no contexto.
```

D.2.6 Exclui Objeto Contexto

São solicitados "nome_do_contexto" e o "nome_do_objeto", e, então, é ativada a rotina correspondente da interface com o banco de dados, que exclui o objeto do contexto (figura 3.11).

Entradas:

```
char *nome_do_contexto;
char *nome_do_objeto;
```

Rotina do BD:

```
EXCLUI_OBJ(ent: ctx, obj);
ctx: contexto;
obj: objeto a ser excluído do contexto.
```

D.2.7 Associa Contexto

São solicitados "nome_do_contexto" e o "nome_do_grupo", e, então, é ativada a rotina correspondente da interface com o banco de dados, que associando um grupo a um contexto, grupo adquire direito de acesso ao contexto (figura 3.11).

Entradas:

```
char *nome_do_contexto;
char *nome_do_grupo;
```

Rotina do BD:

```
AUTORIZA(ent: ctx,gr);
ctx: contexto;
gr: grupo que adquire direito de acesso sobre contexto.
```

D.2.8 Desassocia Contexto

São solicitados "nome_do_contexto" e o "nome_do_grupo", e, então, é ativada a rotina correspondente da interface com o banco de dados, que desassocia um grupo do contexto, o grupo perde direito de acesso ao contexto (figura 3.11).

Entradas:

```
char *nome_do_contexto;
char *nome_do_grupo;
```

Rotina do BD:

```
DESAUTORIZA(ent: ctx,gr);
ctx: contexto;
gr: grupo que perde direito de acesso sobre contexto.
```

D.2.9 Libera Objeto

São solicitados "nome_do_projeto", "nome_da_tarefa" e "nome_do_objeto", e, então, é ativada a rotina correspondente da interface com o banco de dados, que libera objeto do banco de dados privativo (do projetista) para o banco de dados público (figura 3.11).

Entradas:

```
char *nome_do_projeto;     /* igual ao "nome_do_grupo" */
char *nome_da_tarefa;
char *nome_do_objeto;
```

Rotina do BD:

```
LIBERA_OBJ(ent: idp, idu, obj);
idp: transação de projeto;
idu: transação de usuário;
obj: objeto que será liberado para BD público.
```

D.2.10 Inicia Projeto

São solicitados "nome_do_grupo", "nome_do_projeto" e "nome_do_contexto", e, então é ativada a rotina correspondente da interface com o banco de dados, que inicializa uma transação de projeto (figura 3.11).

Entradas:

```
char *nome_do_grupo;
char *nome_do_projeto,     /* igual ao "nome_do_grupo" */
char *nome_do_contexto;
```

Rotina do BD:

```
INICIA_TP(ent: gr,idp,ctx);
gr: grupo ao qual o projetista pertence;
idp: transação de projeto;
ctx: contexto ao qual os objetos do projetista estão
ligados.
```

D.2.11 Finaliza Projeto

É solicitado "nome_do_projeto", e, então, é ativada a rotina correspondente da interface com o banco de dados, que finaliza uma transação de projeto (figura 3.11).

Entradas:

```
char *nome_do_projeto;      /* igual ao "nome_do_grupo" */
```

Rotina do BD:

```
FIM_TP(ent: idp);
```

idp: transação de projeto.

D.2.12 Cancela Projeto

E solicitado "nome_do_projeto", e, então, é ativada a rotina correspondente da interface com o banco de dados, que cancela um projeto em andamento; o banco de dados do projetista (privativo) é removido (figura 3.11).

Entradas:

```
char *nome_do_projeto;      /* igual ao "nome_do_grupo" */
```

Rotina do BD:

```
CANCELA_TP(ent: idp);
```

idp: transação de projeto.

D.3 Administração de Tarefa (figura 3.12)

D.3.1 Inicia Tarefa

São solicitados "nome_do_projeto" e "nome_da_tarefa", e, então é ativada a rotina correspondente da interface com o banco de dados; o projetista informa que iniciou uma tarefa (edição de textos, edição gráfica, compilação ou simulação) (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
```

Rotina do BD:

```
INICIA_TU(ent: idp, idu);
idp: transação de projeto;
idu: transação de usuário.
```

D.3.2 Interrompe Tarefa

São solicitados "nome_do_projeto" e "nome_da_tarefa", e, então é ativada a rotina correspondente da interface com o banco de dados, que permite ao projetista interromper uma tarefa, por um período de tempo (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
```

Rotina do BD:

```
INTERROMPE_TU(ent: idp, idu);
idp: transação de projeto;
idu: transação de usuário.
```

D.3.3 Continua Tarefa

São solicitados "nome_do_projeto" e "nome_da_tarefa", e, então é ativada a rotina correspondente da interface com o banco de dados, que permite que uma tarefa interrompida possa ser recomeçada (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
```

Rotina do BD:

```
CONTINUA_TU(ent: idp, idu);
idp: transação de projeto;
idu: transação de usuário.
```

D.3.4 Finaliza Tarefa

São solicitados "nome_do_projeto" e "nome_da_tarefa", e, então é ativada a rotina correspondente da interface com o banco de dados, que finaliza uma tarefa do projetista (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
```

Rotina do BD:

```
FIM_TU(ent: idp, idu);
idp: transação de projeto;
idu: transação de usuário.
```

D.3.5 Libera Objeto

São solicitados "nome_do_projeto" e "nome_da_tare-

fa" e "nome_do_objeto", e, então é ativada a rotina correspondente da interface com o banco de dados, que libera objeto do banco de dados privativo para o contexto (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
char *nome_do_objeto;
```

Rotina do BD:

```
LIBERA_OBJ(ent: idp, idu, obj);
idp: transação de projeto;
idu: transação de usuário;
obj: objeto que será liberado para o contexto.
```

D.3.6 Empresta Objeto

São solicitados "nome_do_projeto", "nome_da_tarefa" e "nome_do_objeto", e, então é ativada a rotina correspondente da interface com o banco de dados. O objeto fica marcado como passível de empréstimo; a verdadeira transferência é feita através de um "BUSCA_OBJETO". O objeto emprestado não pode ser reemprestado, e deve ser devolvido, através de um "DEVOLVE_OBJETO" (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
char *nome_do_objeto;
```

Rotina do BD:

```
EMPRESTA_OBJ(ent: idp, idu, obj);
idp: transação de projeto;
idu: transação de usuário;
```

obj: objeto que será emprestado para outro projetista.

D.3.7 Transfere Objeto

São solicitados "nome_do_projeto", "nome_da_tarefa" e "nome_do_objeto", e, então é ativada a rotina correspondente da interface com o banco de dados, que autoriza a transferência do objeto para outro projetista. A verdadeira transferência é feita através de um "BUSCA_OBJETO"; este objeto pode ser reemprestado e não precisa ser devolvido (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
char *nome_do_objeto;
```

Rotina do BD:

```
TRANSFERE_OBJ(ent: idp, idu, obj);
idp: transação de projeto;
idu: transação de usuário;
obj: objeto que será transferido para outro projetista.
```

D.3.8 Devolve Objeto

São solicitados "nome_do_projeto", "nome_da_tarefa" e "nome_do_objeto", e, então é ativada a rotina correspondente da interface com o banco de dados, que autoriza a devolução de um objeto que foi emprestado para outro projetista, sendo devolvido através de um "BUSCA_OBJETO" (figura 3.12).

Entradas:

```
char *nome_do_projeto;
char *nome_da_tarefa;
char *nome_do_objeto;
```

Rotina do BD:

DEVOLVE_OBJ(ent: idp ,idu, obj);
idp: transação de projeto;
idu: transação de usuário;
obj: objeto que será devolvido para o projetista.

D.3.9 Busca Objeto

São solicitados "nome_do_projeto", "nome_da_tarefa" e "nome_do_objeto", e, então é ativada a rotina correspondente da interface com o banco de dados que faz a real transferência dos objetos que foram emprestados, devolvidos ou transferidos (figura 3.12).

Entradas:

char *nome_do_projeto;
char *nome_da_tarefa;
char *nome_do_objeto;

Rotina do BD:

BUSCA_OBJ(ent: idp, idu, obj);
idp: transação de projeto;
idu: transação de usuário;
obj: objeto que será transferido.

D.4 Descrição das Funções de Projeto (figura 3.13)

D.4.1 Edição Gráfica

Função de Projeto que permite acesso aos editores gráficos de AMPLO. É exibida uma janela sobreposta com os seguintes editores: **REDES**, para descrição de circuitos compostos por redes de agências, **LAÇO**, a nível de sistema, baseado em redes de Petri, **KAPA**, uma linguagem para descrições estruturais à nível de transferência entre registradores e **NILO**, uma linguagem para descrições estruturais a nível de portas lógicas elementares (figuras 3.13 e 3.14).

D.4.2 Compilação

Ativa o Ambiente de Compilação das descrições textuais. Os compiladores das diversas linguagens estão integrados de forma a terem uma chamada única comum. O próprio compilador reconhece a linguagem na qual uma dada agência está descrita (figura 3.13).

D.4.3 Edição de Textos

Ativa editores de texto como "WordStar", "SideKick" e outros (figura 3.13).

D.4.4 Simulação

Habilita o Ambiente de Simulação, que tem sua própria interface (figura 3.13).

D.5 Descrição das funções de consulta (figura 3.17)

D.5.1 Catálogo

Quando o projetista entrar em LAGO, seu banco de dados privativo estará disponivel para consulta e manipulação, tornando-se a "base de dados corrente". Ele poderá trocar para o banco de dados público ou outro contexto, sem direito a modificação. Esta troca é efetuada através da função "Catálogo" (figura 3.18).

Quando esta opção é apontada, é exibida uma janela sobreposta, com as seguintes funções: "**BD Público**", o banco de dados público será o corrente, os objetos são de domínio público; "**Contexto**", são regras ou lista de objetos que definem um subconjunto na base de dados pública. "**BD Tarefa**", é o banco de dados privativo do projetista.

D.5.2 Apresentação

O projetista pode alterar a forma de visualização dos objetos da base de dados corrente. Pode-se optar pela forma gráfica (árvore) ou textual (lista) (figura 3.17).

D.6 Descrição das Funções de Navegação (figura 3.22)

D.6.1 Alternativas

D.6.1.1 Todas as Alternativas

Função que exibirá uma árvore ou lista contendo todas as alternativas da agência apontada (figura 3.22).

D.6.1.2 Alternativas por Nível

Função que exibirá uma árvore ou lista contendo apenas as alternativas do nível selecionado pelo projetista. O nível corresponde a NILO, KAPA ou LAÇO (figura 3.22).

D.6.1.3 Alternativas por Representação

Função que exibirá uma árvore ou lista contendo apenas as alternativas da representação selecionada pelo projetista, ou seja, alternativas que foram descritas na representação selecionada: gráfica (G), textual (T) ou gráfica e textual (GT) (figura 3.22).

D.6.1.4 Última Alternativa Absoluta

Função que exibirá uma árvore ou lista contendo apenas a última alternativa absoluta, ou seja, a última alternativa criada dentre todas as da agência apontada (figura 3.22).

D.6.1.5 Última Alternativa por Nível

Função que exibirá uma árvore ou lista contendo apenas as últimas alternativas por nível (NILO, KAPA ou LAÇO) da agência apontada (figura 3.22).

D.6.2 Versões

D.6.2.1 Versões por Usos

Função que exibirá uma árvore ou lista contendo todas as versões REDES onde a alternativa é usada como ocorrência (figuras 3.23 e 3.24).

D.6.2.2 Versões por Linguagem

Função que exibirá uma árvore ou lista contendo todas as versões dependendo da linguagem utilizada para defini-las: NILO, KAPA, LAÇO ou REDES (figuras 3.22 e 3.23).

D.6.2.3 Versões por Representação

Função que exibirá uma árvore ou lista contendo apenas as versões dada a representação selecionada pelo projetista, ou seja, últimas versões gráficas, últimas versões textuais, últimas gráficas e textuais (figuras 3.22 e 3.23).

D.6.2.4 Última Versão Absoluta

Função que exibirá uma árvore ou lista contendo a última versão da alternativa apontada (figura 3.22).

D.6.2.5 Última Versão por Nível

Função que exibirá uma árvore ou lista contendo apenas as últimas versões por nível (NILO, KAPA ou LAÇO) da alternativa apontada (figuras 3.22 e 3.23).

D.6.3 Composição

D.6.3.1 Composição

Função que exibirá uma árvore ou lista contendo as

ocorrências presentes na versão composta (REDES) apontada (figura 3.24).

D.6.3.2 Usos

Função que exibirá uma árvore ou lista contendo as versões REDES, da versão composta apontada (figuras 3.23 e 3.24).

D.6.3.3 Composição por Linguagem

Função que exibirá uma árvore ou lista contendo a árvore de composição da versão composta apontada, dependendo da linguagem escolhida (REDES, KAPA, LAÇO ou NILO) (figura 3.24).

D.6.3.4 Composição por Representação

Função que exibirá uma árvore ou lista contendo as versões / alternativas da versão composta apontada, dependendo da representação (gráfica (G), textual (T) ou gráfica e textual (GT)) (figura 3.24).

D.6.3.5 Composição por Classe

Função que exibirá uma árvore ou lista contendo as ocorrências de alternativas ou versões, se foi apontada uma versão composta (figura 3.24).

D.7 Saída (figura 3.19)

D.7.1 Sistema

Saída temporária ao sistema operacional. No DOS digitando "EXIT" o projetista poderá retornar ao ambiente AMPLO normalmente (figura 3.19).

D.7.2 Término

Saída definitiva de AMPLO, o usuário não poderá retornar, a não ser que execute novamente a chamada de LAGO (figura 3.19).

BIBLIOGRAFIA

[BAG 89a] BAGGIO, André Uma Interface de Gerenciamento para o projeto Tranca. Porto Alegre, CPGCC da UFRGS, 1989.

[BAG 89b] BAGGIO, André GIM: Um Gerenciador de interfaces orientadas a Menus Trabalho submetido a 18 avas Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, Argentina, Septiembre 1989.

[BAN 86] BANCILHON, F. et al. A Model of CAD Transaction. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 11., August 21 - 23, 1986, Stockholm, Proceedings... Stockholm, IBM, 1986. p. 25-33.

[BUS 86] BUSHNELL, M.L. & DIRECTOR, S.W. VLSI CAD Tool Integration using the ULYSSES Environment. In: DESIGN AUTOMATION CONFERENCE, 23., June 29 - July 2, 1986, Las Vegas. Proccedings... New York, ACM/IEEE, 1986. p.55-61.

[CZE 90] CZEJDO, B., ELMASRI, R., RUSINKIEWICZ, M. & EMBLEY, D.W. A Graphical data manipulation language for an extended entity-relationship model. Computer, Los Alamitos, V.23, n.3, p.26-36, March 1990.

[DAT 77] DATE, C. J. An Introduction to Database Systems. Readings, Addison-Wesley, 1977.

[DEL 86] DELISLE, Norman & SCHWARTZ, Mayer - Neptune: a Hypertext System for CAD Applications. SIGMOD RECORD, New York, V.15, n.2, p.28-30, June 1986. Trabalho apresentado no ACM SIGMOD'86.

[DIT 86] DITTRICH, K. Controled Cooperation in Engineering Database Systems. San Jose, IBM Research Report, 1986.

[DON 86] DONAHUE, James & WIDOM, Jennifer Whiteboards: Graphical Database Tool. ACM Transactions on Office Systems, New York, V.4, n.1, p.24-31, Jan. 1986.

[FOL 82] FOLEY, J.D. & VAN DAN, A. Fundamentals of interactive computer graphics. Reading, Addison-Wesley, 1982.

[FRE 88] FREITAS, Carla M.D.S; GOLENDZINER, L.G.; WAGNER, F.R. O sistema AMPLO: Um Ambiente Integrado para Projeto de Sistemas Digitais. In: SEMINARIO INTEGRADO DE SOFTWARE E HARDWARE, 15, 17 - 22 de jul. 1988, Anais... Rio de Janeiro, SBC, 1988. p.272-284.

[GED 88] GEDYE, David & KATZ, Randy Browsing the Chip Design Database. In: DESIGN AUTOMATION CONFERENCE, 25. June 12-15, 1988, Anaheim. Proccedings... New York, ACM, 1988. p. 269-274.

[GIR 87] GIRCZYC, E. & STEM, T. Ly. An IC Design Environment Based on the Smalltalk Model-View-Controller Cosntruct. In: DESIGN AUTOMATION CONFERENCE, 24., 1987, Miami Beach. Proceedings ... New York, ACM/IEEE, 1987. p.757-763.

[GOL 84] GOLDBERG, A. Smalltalk-80: Readings, The Interactive Programming Environment. Addison-Wesley, 1984.

[GOL 88] GOLENDZINER, L.G. & BECKER, K. Interface orientada a objetos para um ambiente de CAD de sistemas digitais. In: SIMPOSIO BRASILEIRO DE BANCO DE DADOS, 3., 24-25 março 1988, Recife. Anais ... Recife, SBC/UFPe, 1988. pp. 213-226.

[HAR 83] HARDER., T. & REUTER, A. Principles of Transaction-Oriented Database Recovery. Computing Surveys. New York, v.15, n.4, p.287-317, dec. 1983.

[HAR 88] HARDER., T. et al. Coupling Engineering Workstation to a Database Server. In: CONFERENCE ON DATA AND KNOWLEDGE SYSTEMS FOR ENGINEERING AND MANUFACTORING, 3., oct 19-20, 1987, Hartford, Proceedings... [S.l. : s.n.] 1987.

[HAR 89] HARDER., T. Engineering Aplications - A Chalenge for Next Generation of DBMS. Kaiserslautern, [S.n.] 1989.

[KNA 86] KNAPP, D.W & PARKER, A.C. A Design Utility Manager: the ADAM Planning Engine. In: DESIGN AUTOMATION CONFERENCE, 23., June 29 - July 2, 1986, Las Vegas. Proccedings... New York, ACM/IEEE, 1986. p.48 - 54.

[LAE 89] LAENENS, E., STAES, F. & VERMEIR, D. Browsing à la carte in object-oriented Databases. The Computer Journal, New York, V.32, n.4, p.333-340, aug, 1989.

[MIL 89] MILLER, J.; GRONING. K.; SCHULZ. G.; WHITE. C., The Object-Oriented Integration Methodology of Cadlab Work Station Design Environment. In: DESIGN AUTOMATION CONFERENCE, 26., Aug. 1989. Proceedings... New York: ACM/IEEE, 1989.

[NET 90] NETO, M.M.O. Um Estudo sobre Transações de Projeto e sua aplicação no Sistema AMPLO. Porto Alegre, PGCC da UFRGS, 1990.

[OLA 87a] OLABARRIAGA, S.D.; PINHO, M.S.; COMBA, J.L.D Interface de saída com dispositivos gráficos. Porto Alegre, PGCC da UFRGS, 1987. (Relatório de Pesquisa, 79).

[OLA 87b] OLABARRIAGA, S.D. Interface com dispositivos de entrada gráfica. Porto Alegre, PGCC da UFRGS, 1987. (Relatório de Pesquisa, 83).

[OLA 87c] OLABARRIAGA, S.D., & COMBA, J.L.D Extensão da interface de saída com dispositivos gráficos. Porto Alegre, PGCC da UFRGS, 1987. (Relatório de Pesquisa, 84).

[OLA 88] OLABARRIAGA, S.D.; PINHO, M.S.; COMBA, J.L.D; FREITAS, C.M.D.S. Ferramentas gráficas do sistema AMPLO. In: SIMPOSIO BRASILEIRO DE COMPUTAÇÃO GRAFICA e PROCESSAMENTO DE IMAGENS, 1., 19-20 abr. 1988, Petrópolis. Anais... Rio de Janeiro, SBC/COPPE, 1988. p.78-87.

[OLA 89] OLABARRIAGA, S.D., COPSTEIN, B. & FREITAS, C.M.D.S Ferramentas para especificação e controle da interface com o usuário. Porto Alegre, PGCC da UFRGS, 1989. (Relatório de Pesquisa, 105).

[PIN 88] PINHO, M.S., OLABARRIAGA, S.D. & COMBA, J.L.D. Pacote Gráfico para Editores do Sistema AMPLO. Porto Alegre, PGCC da UFRGS, de 1988. (Relatório de Pesquisa, 102).

[SHN 83] SHNEIDERMAN, B. Human Factors of Interaction Software. In: SCIENTIFIC SYMPOSIUM OF INTERATION SCIENCE CENTER OF IBM. March 18, 1983, Germany. Proceedings... [S.l. : s.n.] 1983.

[SHN 87] SHNEIDERMAN, B. "Designing the User Interface." Readings, Mass., Addison-Wesley, 1987.

[SIL 88] SILVA FILHO, J.F., WAGNER,F.R. & LE FAOU, C. LAÇO - Uma Linguagem para descrição de Hardware no nível de Sistema. Porto Alegre, PGCC, UFRGS, Outubro/1988. (Relatório de Pesquisa 94).

[SMI 89] SMITH. W.D., DUFF. D., DRAGOMIRECKY. M., CALDWELL. J., HARTMAN. M., JASICA. J. & D'ABREU. M.A., FACE Core Environment: The Model and its Application in CAE/CAD Tool Development. In: DESIGN AUTOMATION CONFERENCE, 26., August 1989. Proceedings ... New York: ACM/IEEE, 1989.

[TEI 84] TEITELMAN, W. A tour through Cedar. IEEE Software, Los Alamitos, V.1, n.2, p.44-73, apr. 1984.

[WAG 86a] WAGNER, F.R., FREITAS, C.M.D.S & GOLENDZINER, L.G. Equivalência de Descrições Textuais e Gráficas de Sistemas Digitais num Ambiente de CAD. SEMINARIO INTEGRADO DE SOFTWARE E HARDWARE, 13., 19 - 25 jul., 1986, Olinda. Anais ... Olinda, SBC, 1986. p. 486 - 493.

[WAG 86b] WAGNER, F. R. A Multi-level Digital Systems Simulator based on Nets of Agencies. In: JSST CONFERENCE ON RECENT ADVANCES IN SIMULATION OF COMPLEX SYSTEMS. 15-17 July, 1986, Tokio. Proceedings... Tokyo, JSST, 1986. p. 125-130.

[WAG 87a] WAGNER, F.R.; SASSO-FREITAS C.M.; GOLENZINER, L.G. Linguagens de descrição de hardware para suporte a integração do processo de projeto em AMPLO. Porto Alegre, PGCC da UFRGS, 1987. (Relatório de Pesquisa, 65).

[WAG 87b] WAGNER, F.R., SASSO-FREITAS, C.M. NILO - uma linguagem de descrição de hardware no nivel de portas lógicas. Porto Alegre, PGCC da UFRGS, 1987. (Relatório de Pesquisa, 66).

[WAG 87c] WAGNER, F.R. KAPA - uma linguagem de descrição de hardware no nivel de transferência entre registradores. Porto Alegre, PGCC da UFRGS, 1987. (Relatório de Pesquisa, 68).

[WAG 87d] WAGNER, F.R.; SASSO-FREITAS C.M.; GOLENZINER, L.G. A digital systems design methodology based on nets of agencies. In: IFIP WG10.2 INTERNATIONAL CONFERENCE ON COMPUTER HARDWARE DESCRIPTION LANGUAGES AND THEIR APPLICATIONS, 8., Apr. 27-29, 1987, Amsterdam. Proccedings... Amsterdam. North-Holland, 1987. p.213-224.

[WAG 88a] WAGNER, F.R.; FREITAS, C.M.D.S; GOLENDZINER, L.G. The AMPLO system - An Integrated Environment for digital Systems Design. In: WORKSHOP ON TOOL INTEGRATION AND DESIGN ENVIRONMENTS, 26-27 Nov, 1987, Paderborn. Proccedings... Amsterdan, North-Holland, 1988. p.221-232.

[WAG 88b] WAGNER, F.R. Ambientes Integrados de Projeto de circuitos VLSI. In: SIMPOSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 3., 13-15 abr., 1988, Gramado. Anais Porto Alegre, SBC, 1988. p.147-160.

[WAL 89] WALTER, M. Projeto Detalhado do Editor Gráfico KAPA. Porto Alegre, PGCC da UFRGS, 1989.

[WAP 88] WAGNER, P.R. Editor gráfico REDES. Porto ALegre, CGPCC da UFRGS, 1988.

[WEB 88] WEBER, R. F. e WEBER, T.S. Uso de ambientes gráficos em estações de trabalho. In: SIMPOSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 3., 13-15 abr., 1988, Gramado. Anais... Porto Alegre, SBC, 1988. p.161-171.

[WOL 86] WOLF, W. An Object-Oriented, Procedural Database for VLSI Chip Planning. In: DESIGN AUTOMATION CONFERENCE, 23., june 29 - july 2, 1986, Las Vegas. Proceedings... New York, ACM/IEEE, 1986. p.744-751.

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

"LAGO - Linguagem de Acesso Global ao Sistema AMPLO"

Dissertação apresentada aos Srs.

Prof. Dr. Arnaldo Hilário Viegas de Lima (IBM/RJ)

Profa. Carla Maria Dal Sasso Freitas

Prof. Dr. Cirano Iochpe

Prof. Dr. José Palazzo Moreira de Oliveira

Visto e permitida a impressão

Porto Alegre, 05 / 02 / 92

Prof. Dr. Flávio Rech Wagner
Orientador

Prof. Dr. Ricardo Augusto da L. Reis
Coordenador do Curso de Pós-Graduação
em Ciência da Computação

ESR/LJSJr

\DIVCPGCC\ASSINA.BAN