FEGRES: FErramentas GRáficas para Engenharia de Software

Guilherme Horta Travassos

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Ana Regina Cavalcanti da Rocha, D.Sc.
(Presidente)

Prof. Jano Moreira de Souza, Ph.D.
(co-orientador)

Prof. Antônio Carlos dos Santos, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

JANEIRO DE 1990

TRAVASSOS, GUILHERME HORTA

FEGRES: FErramentas GRáficas para Engenharia de Software [Rio de Janeiro] 1990

XI, 197 p., 29,7cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1990)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Ferramentas para Engenharia de Software I. COPPE/UFRJ II. Título (série)

Para Liliam, Caio, Miriam e Gustavo

AGRADECIMENTOS

À Liliam, pela sua paciência, amor e compreenssão.

Aos meus pais, pelo incentivo e o carinho que propiciaram.

Ao meu irmão, pelo estímulo e a compreenssão.

À Prof. Ana Regina Cavalcanti da Rocha pela orientação e acolhida na sua linha de pesquisa.

Ao Prof. Jano Moreira de Souza pela sua co-orientação e as idéias geniais.

Ao Luiz Carlos Monte pelo suporte na área de Interface com o Usuário e diversas sugestões construtivas.

Ao Cláudio Trotta pelo suporte em Banco de Dados e o estímulo constantes.

Ao Washington Lucena pela participação na implementação e depuração de alguns módulos da ferramenta.

Ao Departamento de Circuitos Elétricos da Universidade Federal de Juiz de Fora, particularmente nas pessoas dos Professores Herculano Coimbra Filho e Paulo Roberto de Castro Villela, pelo incentivo e a oportunidade concedida.

A todos os meus amigos e parentes por compreenderem as minhas constantes ausências e me estimularem sempre, em especial para minha tia Eleni pelo apartamento emprestado.

Aos funcionários e alunos da COPPE/SISTEMAS que propiciaram um ambiente adequado ao desenvolvimento deste trabalho.

À CAPES pelo apoio financeiro.

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

FEGRES: FErramentas GRáficas para Engenharia de Software

Guilherme Horta Travassos

Janeiro de 1990

Orientador: Ana Regina Cavalcanti da Rocha, D.Sc.

Co-Orientador: Jano Moreira de Souza, PhD.

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta um conjunto de ferramentas automatizadas, FEGRES, para uso na especificação de produtos de software. As ferramentas foram desenvolvidas de forma a manter a documentação o mais consistente possível a partir da integração das informações geradas por cada ferramenta e armazenadas por um sistema gerenciador de dados especialmente desenvolvido para FEGRES. A interface com o usuário é construída a partir de um sistema gerenciador de interface que mantém a consistência de comunicação entre as ferramentas . FEGRES implementa os métodos da Análise Estruturada e Modelo de Entidades-Relacionamentos.

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

FEGRES: Graphics Tools for Software Engineering

Guilherme Horta Travassos

January, 1990

Thesis Supervisor: Ana Regina Cavalcante da Rocha, D.Sc.

Thesis Co-Supervisor: Jano Moreira de Souza, PhD.

Department: Computer Systems Enginnering

This work presents an automated tool's set, FEGRES, for the specification of software products. The tools were developed in order to mantain the documentation as consistent as possible according to the integrated information created by the tools and stored by a data management system specially developed for FEGRES. The user interface keeps the consistency of the communication between the tools and was built using an interface management system package. FEGRES suports the Structured Analysis Method and the Entity-Relationship Model.

ÍNDICE

I. INTRODUÇÃO

I.1 Motivação 1

I.2 Ferramentas Disponíveis no Mercado para Especificação de Sistemas Utilizando Técnicas Estruturadas 7

I.2.1 Ferramentas Disponíveis no Exterior 7

I.2.2 Ferramentas Disponíveis no Brasil 10

I.2.3 Ferramentas Desenvolvidas na COPPE 11

I.3 FEGRES: uma FErramenta GRáfica para Engenharia de Software 12

I.4 Organização da Tese 14

II. A ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE USANDO ANÁLISE ESTRUTURADA

II.1 A Fase de Análise 15

II.1.1 Análise Estruturada 17

II.2 Diagrama de Fluxo de Dados 19

II.2.1 Descrição Geral 19

II.2.2 Elementos de um DFD 19

II.3 Dicionário de Dados 24

II.3.1 Descrição Geral 24

II.3.2 Elementos Descritos. 24

II.4 Descrição da Lógica dos Processos. 29

II.4.1 Descrição Geral. 29

II.5 Diagrama de Entidades-Relacionamentos 30

II.5.1 Descrição Geral. 30

II.5.2 Elementos de um DER. 30

II.5.3 Extensões Utilizadas. 33

II.6 Diagramas de Transição de Estados 34

II.6.1 Descrição Geral. 34

II.6.2 Elementos de um DTE. 34

II.7 Justificativas para Extensões ao Método Análise Estruturada 36

III. FEGRES: UM CONJUNTO DE FERRAMENTAS GRÁFICAS PARA ENGENHARIA DE SOFTWARE

III.1 Introdução 39

III.2 Especificação da Ferramenta. 40

III.2.1 Requisitos Funcionais 40

III.2.2 Requisitos de Interface. 42

III.2.3 Requisitos de Qualidade. 53

III.2.4 Requisitos de Desempenho 54

III.2.5 Requisitos de Armazenamento de Informações. 55

III.2.6 Prioridades de Implementação. 57

IV. O PROJETO DE FEGRES

IV.1 Introdução 59

IV.2 Recursos Necessários. 60

IV.2.1 Hardware. 60

IV.2.2 Software. 60

IV.3 Projeto Físico dos Objetos 64

IV.3.1 Objetos Básicos Tratados por FEGRES. 64

IV.3.2 Restrições de Integridade dos Objetos. 66

IV.3.3 Tabelas para Representação dos Objetos. 69

IV.3.4 Principais Algoritimos para Tratamento dos Objetos. 86

V. CONCLUSÕES. 117

REFERÊNCIAS BIBLIOGRAFICAS 119

APÊNDICE A: MANUAL DO USUÁRIO DE FEGRES 124

Índice de Figuras

I.1- Visão Simplificada do Ambiente FEGRES 13

II.1- Um DFD com a explosão de um processo 20

II.2- Representação Gráfica dos Elementos do DFD 21

II.3- Representação de Entidades Externas Repetidas 21

II.4- Representação de um Processo no DFD 22

II.5- Representação de Depósitos de Dados Repetidos num DFD 23

II.6- Representação de Fluxo de Dados no DFD 23

II.7- Ficha para Descrição de Entidade Externa 25

II.8- Ficha para Descrição de Processo 25

II.9- Ficha para Descrição de Depósito de Dados 26

II.10- Ficha para Descrição de Fluxo de Dados 26

II.11- Ficha para Descrição de Estrutura de Dados 27

II.12- Ficha para Descrição de Elemento de Dados 27

II.13- Ficha para Descrição de Ítem de Glossário 28

II.14- Representação de Entidade e Relacionamento 31

II.15- Diagrama de Entidade-Relacionamento 32

II.16- Representação dos Elementos de um DTE 35

II.17- Diagrama de Transição de Estados 36

III.1 Estágio Atual de FEGRES 40

III.2 DFD Nível Contexto 43

III.3 DFD Nível 1 44

III.4 DFD Explosão do Processo 1 45

III.5 DFD Explosão do Processo 2 46

III.6 DFD Explosão do Processo 3 47

III.7 DFD Explosão do Processo 4 48

III.8 DFD Explosão do Processo 5 50

IV.1 O Modelo de Dados de FEGRES 65

A.1 Aparência de Janelas Sobrepostas 151

A.2 Componentes de uma Janela 153

A.3 Ícones 154

A.4 Caixa de Mensagem 155

A.5 Caixa de Diálogo 156

A.6 A Máquina Virtual WINDOWS 158

A.7 Filas de Mensagens no WINDOWS 160

A.8 Uma Janela de Aplicação de FEGRES 165

A.9 Caixa de Diálogo Abrir Projeto 166

A.10 Caixa de Diálogo para Atributos de Projeto 167

A.11 Caixa de Diálogo Salvar Como 168

A.12 Caixa de Diálogo para Impressão 168

A.13 Objetos Representados por FEGRES 170

A.14 Ferramenta com Opção GRID 171

A.15 Ferramenta com Opção ZOOM 172

A.16 Janela para Escolha do Percurso 172

A.17 Ficha de Descrição de Processo 180

A.18 Ficha de Descrição de Entidade Externa 183

A.19 Ficha de Descrição de Depósito de Dados 184

A.20 Ficha de Descrição de Fluxo de Dados 186

A.21 Ficha de Descrição de Estrutura de Dados 188

A.22 Ficha de Descrição de Elemento de Dados 190

A.23 Ficha de Descrição de Ítem de Glossário 192

A.24 Ficha de Descrição de Entidade 194

A.25 Ficha de Descrição de Relacionamento 195

A.26 Ficha de Descrição de Ligação 196

A.27 Ficha de Descrição de Agregação 197

CAPÍTULO I

INTRODUÇÃO

I.1 Motivação

A crescente evolução dos sistemas computacionais tem provocado o desenvolvimento de produtos de software capazes de realizar tarefas até então impensáveis. A construção destes produtos envolve a utilização de métodos adequados no sentido de se manter um nível de qualidade e custo de desenvolvimetno apropriado. Não podemos pensar em desenvolver produtos de software apenas escrevendo linhas de código numa determinada linguagem de programação. Como na construção de uma obra, onde o engenheiro tem que se preocupar em realizar todo o projeto antes de começar a edificá-la, o desenvolvimento de um produto de software exige do engenheiro de software a utilização de métodos apropriados que garantam a qualidade do produto e o seu desenvolvimento dentro dos custos e tempo previstos.

Nos estágios iniciais da utilização do computador pelo homem o desenvolvimento de produtos de software consistia basicamente de programação, sendo esta atividade executada na maioria das vezes por programadores individuais para resolução de problemas matemáticos,HAUSEN [1]. Atualmente o desenvolvimento de um produto envolve não apenas programação, mas uma série de atividades complementares e essenciais, que demandam tecnologias apropriadas e que proporcionem produtos com qualidade desejada desenvolvidos dentro do tempo e custos previstos.

Para podermos controlar o processo de desenvolvimento de produtos de software alguns métodos têm sido criados com o objetivo de aumentar a qualidade e a produtividade dos projetos. Esses métodos muitas vezes tornam o desenvolvimento de um projeto uma tarefa árdua devido ao volume de documentos produzidos, e também por não apoiarem todo o ciclo de

desenvolvimento do software, o que causa o aumento da complexidade de gerenciamento. . Com o intuito de se obter um controle perfeito do processo de desenvolvimento de um produto de software, engenheiros de software têm se preocupado em definir Ambientes de Desenvolvimento de Software que permitam controlar todas as informações geradas no desenrolar do projeto, independente de sua forma e apresentação.

Ambientes de Desenvolvimento de Software, ou Ambientes para Engenharia de Software, têm sido determinado como um bom meio de aumentar a produtividade através do deslocamento dos projetos do mundo do papel para o mundo automatizado, PENEDO [2]. Em seu estágio inicial estes ambientes se restringiam a uma coleção de ferramentas, que implementavam no computador um conjunto de convenções notacionais e operações possíveis de serem automatizados, pertencentes a um determinado método, isoladas entre si. Isto quer dizer que não existia uma integração entre essas ferramentas, pois na maioria dos casos, além de não se comunicarem umas com as outras, executavam também em máquinas diferentes tornando muito difícil o processo de integração e comunicação entre elas. NUNES [3]

Após várias tentativas para integração das ferramentas, verificou-se que seria necessária uma integração sintática e funcional das mesmas para que pudessem melhorar as caracterísitcas até então existentes nos ambientes. A integração sintática permite que as ferramentas troquem informações entre si, através do compartilhamento de uma base de dados comum ao passo que a integração funcional permite que as ferramentas executem num mesmo equipamento. A integração das ferramentas realizada desta forma possibilita a adoção de um conjunto de regras de gerenciamento e procedimentos para suportar trocas, controle e distribuição de informação de maneira uniforme e consistente. PENEDO [2]

DART et alli [4] classificam ambientes de desenvolvimento de software em quatro classes distintas que são identificadas por: ambientes centralizados em linguagem, ambientes orientados a estrutura, ambientes orientados a conjunto de ferramentas e ambientes baseados em métodos.

Ambientes centralizados em linguagens proporcionam ao desenvolvedor da aplicação um conjunto de ferramentas para suporte a uma determinada linguagem. São altamente interativos e não se prestam bem ao desenvolvimento de sistemas muito complexos. Funcionam bem para pequenas aplicações, onde o nível de gerenciamento é baixo. Este tipo de ambiente encoraja o desenvolvedor da aplicação a adotar um estilo exploratório de programação a fim de alcançar um alto nível de produtividade. Se prestam muito bem a fase de programação do ciclo de desenvolvimento. Exemplos destes ambientes são Interlisp para linguagem LISP, TEITELMAN [5], e Smalltalk para linguagem Smalltalk, GOLDBERG [6].

Ambientes orientados a estrutura permitem ao desenvolvedor da aplicação manipular as estruturas diretamente e obter uma múltipla visão textual do programa. São independentes de linguagem e se prestam muito bem a serem geradores para ambientes orientados a estrutura. Este tipo de ambiente tem sido muito usado na fase de programação do ciclo de desenvolvimento de sistemas que não sejam muito complexos. Exemplos deste ambiente são Aloe, FEILER [7], e Pecan, REISS [8].

Ambientes orientados a conjunto de ferramentas consistem de uma coleção de pequenas ferramentas desenvolvidas primariamente para suportar a fase de programação do ciclo de desenvolvimento. Não exercem controle algum sobre o uso destas ferramentas que podem ser utilizadas para o desenvolvimetno de sistemas complexos. Apesar de serem muito populares devido a portabilidade e linearidade de suas ferramentas, este tipo de ambiente não consegue controlar a manutenção de sistemas complexos. Exemplos destes ambientes são Unix/PWB, DOLOTTA [9], e Arcadia, TAYLOR [10].

Ambientes orientados a método suportam um particular método para desenvolvimento de produtos de software. Este método pode ser para uma fase particular do ciclo de desenvolvimento ou então pode gerenciar todo o processo de desenvolvimento, independente do grau de formalismo adotado. Métodos semiformais tem sido muito utilizados para construir ambientes deste tipo, através da construção de ferramentas isoladas , que permitem seu uso por apenas um

usuário de cada vez. Alguns exemplos de ambientes orientados a método, segundo DART et alli [4], são Diagramas de Entidade-Relacionamento, CHEN [11], PSL/PSA, TEICHROEW [12], SADT, ROSS [13], e Análise Estruturada de Sistemas, GANE [14].

A partir destes ambientes é possível ao engenheiro de software construir uma aplicação com maior qualidade e produtividade. Dentre estes ambientes os que mais se destacam são os que fazem uso de métodos estruturados para desenvolvimento. Uma justificativa para este fato é que o uso dos métodos estruturados de forma manual causa problemas para o engenheiro de software. Devido a utilização de papel e lápis para a construção dos documentos, alguns problemas ocorrem para o desenvolvedor, dos quais podemos destacar que as alterações e correções na documentação tornam-se trabalhosas; as regras de formação dos documentos podem ser desobedecidas; a obtenção de informações é feita de maneira lenta e através do acesso a vários arquivos; ocorre um grande número de redundâncias das informações entre os documentos , nomes diferentes acabam representando os mesmos objetivos, além de as vezes ocorrer a ausência de informações. Além destes ainda temos a ocorrência de uma ausência de validação e uma pequena capacidade de verificação devido a não utilização de uma linguagem formal, AGUIAR [15] NOGUEIRA [16]. O uso de técnicas estruturadas no desenvolvimento de produtos de software envolve a produção de vários documentos, alguns deles apresentados em forma gráfica, e que, apesar de proporcionarem um aumento da qualidade e confiabilidade do software produzido, levam os analistas e projetistas a dispenderem um considerável percentual do tempo de desenvolvimento na sua elaboração, manutenção e deteção de inconsistências que por ventura possam existir. Segundo YOURDON [17] , uma nova restrição tecnológica tem se tornado comum: muitos programadores e analistas carecem de ferramentas automatizadas que tornem possível desenvolver e manter os modelos gráficos de análise e projeto estruturados eficientemente.

Com o objetivo de suprir o engenheiro de software de ferramentas automatizadas que aumentassem a produtividade e melhorassem a qualidade do desenvolvimento de um produto de software, ROCHA et alli [18] propuseram a construção de um conjunto de ferramentas automati-

zadas, implementadas em microcomputador, para apoio ao desenvolvimento de software, com os seguintes objetivos :

.facilitar alterações na documentação;

.permitir o armazenamento de todas as informações geradas durante o desenvolvimento;

.permitir flexibilidade e fácil acesso a informações complexas;

.permitir que as regras pré-definidas de formação dos documentos sejam obedecidas;

.verificar e manter a consistência de toda a documentação;

.permitir que informações a respeito do produto de software sejam obtidas mais rapidamente;

.manter as informações o mais atualizadas possível;

.permitir que diferentes usuários , mesmo aqueles que não possuam experiência na área de computação, tenham acesso ao sistema através de uma interface com o usuário simples e consistente;

.aumentar a produtividade do Engenheiro de Software;

.aumentar a qualidade dos produtos desenvolvidos.

Para atingir estes objetivos foi previsto, inicialmente, o desenvolvimento das seguintes ferramentas :

.Editor de Diagrama de Fluxo de Dados, cujo objetivo é criar e modificar diagramas de fluxo de dados. Possibilita, também, a listagem dos diagramas criados;

.Gerador de Dicionário de Dados, cujo objetivo é armazenar, organizar e possibilitar a disponibilidade das informações sobre os dados utilizados no processo de desenvolvimento e sobre os módulos do projeto;

.Editor de Lógica de Processos, cujo objetivo é criar e modificar a lógica dos processos e emiti-los quando for requerido;

.Editor de Diagramas de Acesso Imediato a Dados, cujo objetivo é criar e modificar diagramas de acesso imediato aos dados. Possibilita, também, a listagem dos diagramas criados;

.Avaliador de Diagrama de Fluxo de Dados, cujo objetivo é avaliar se as regras de sintaxe da análise estruturada foram obedecidas na construção dos diagramas de fluxo de dados de um sistema;

.Avaliador de Processos, cujo objetivo é avaliar se as regras de sintaxe da análise estruturada foram obedecidas na construção da lógica dos processos de um sistema;

.Avaliador de Diagramas de Acesso Imediato a Dados, cujo objetivo é avaliar se as regras de sintaxe da análise estruturada foram obedecidas na construção do diagrama de acesso imediato de um sistema;

.Verificador de Consistência, cujo objetivo é verificar as possíveis contradições entre as informações geradas;

.Editor de Gráfico de Estrutura, cujo objetivo é criar, editar e modificar os gráficos de estruturas modulares;

.Avaliador de Gráfico de Estrutura, cujo objetivo é analisar a qualidade do gráfico de estrutura à luz das normas de projeto estruturado;

.Editor de Pseudocódigo, cujo objetivo é criar, editar e modificar a descrição dos módulos em pseudo-código.

Destas ferramentas foram implementadas o editor de fluxo de dados, AGUIAR [15], o Dicionário de Dados, BLASCHEK [19], e o editor de Gráfico de Estrutura NOGUEIRA [16]. Posteriormente foram construídas ainda ferramentas para apoio a documentação, PRODOC,

VALLE [20], para realizar estimativas de custo, ESTIME, MENEZES [21], e ferramentas para desenvolvimento de aplicações para tempo real, Editor para o método DARTS, CAVALCANTE [22] e um dicionário de dados adaptado ao mesmo, DDTR, ULLOA [23].

I.2 Ferramentas Disponíveis no Mercado para Especificação de Sistemas Utilizando Técnicas Estruturadas

O desenvolvimento de ferramentas de apoio ao engenheiro de software vem ocorrendo há algum tempo. Encontramos disponíveis em centros de pesquisa e no mercado vários modelos de ferramentas que podem ser usadas pelo engenheiro de software na construção de um produto. Estas ferramentas suportam determinadas fases do ciclo de desenvolvimento e , algumas vezes, são ferramentas isoladas, não permitindo que o projeto seja executado de maneira única e consistente.

A seguir algumas ferramentas disponíveis no mercado são apresentadas. Esta lista não tem a intenção de ser completa, mas sim, de apresentar a tendência do mercado e as principais ferramentas disponíveis.

I.2.1 Ferramentas Disponíveis no Exterior

1) Analyst/RT: utilizada para especificação estruturada de sistemas, cria , verifica , e documenta especificações para sistemas de tempo real usando o método de Hatley e diagramas de Ward/Mellor estendidos para a análise estruturada de Yourdon/DeMarco. Está disponível para sistemas VAX/VMS, incluindo estações de trabalho VAX e Apollo. Foi desenvolvida por *Mentor Graphics*, 8500 SW Creekside Place, Beaverton, OR 97005;

2) Anatool: utilizada para prototipação e desenvolvimento de sistemas atráves da análise estruturada. Está disponível para equipamentos Macintosh. Foi desenvolvida por *Advanced Logi-*

cal Software, 9903 Santa Monica Blvd.,Ste. 108, Beverly Hills, CA 90212;

3) AutoDraw: utilizada para prototipação e desenvolvimento de sistemas através do desenho de diagramas de Entidades-Relacionamentos a partir de descrições textuais. Está disponível para equipamento PC compatível. Foi desenvolvida por *Chen and Associates Inc.*, 4884 Constitution Ave., Ste. 1E, Baton Rouge, LA 70808;

4) Brackets: utilizada para projeto de sistemas através da utilização de projeto estruturado e codificação, gerando código em linguagem COBOL. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Optima Inc.*, 1300 Woodfield Rd., Ste. 400, Schaumburg, IL 60173;

5) CASE 2000 (Design Aid/Data Model Option/Real Time Option/LCM): uma ferramenta integrada para análise estruturada de sistemas, projeto, e documentação para desenvolvimento e prototipação de sistemas. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Nastec Corp.*, 24681 Northwestern Hwy., Southfield, MI 48075;

6) DFDdraw/SCdraw: produz diagramas gráficos para análise estruturada de sistemas e projeto estruturado de sistemas. Pode ser usada também para engenharia reversa. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *McDonnell Douglas*, ISC, P.O. box 516, Dept. L863, Bldg. 280-L2, St. Louis, MO 63166;

7) Deft: utilizada para manutenção e projeto de sistemas através de documentação, projeto de bases de dados e documentação. Está disponível para equipamentos Macintosh e VAX. Foi desenvolvida por *Deft*, 557 Dixon Rd., Ste. 110, Rexdale, Ont., Canada M9W 1H7;

8) DesignAid: utilizada para desenvolvimento, manutenção e prototipação de sistemas através de métodos de análise e projeto estruturados. Está disponível para equipamentos PC compatíveis suportando aplicações em rede. Foi desenvolvida por *Nastec Corp.*, 24681 North-

western Hwy., Southfield, MI 48075;

9) Designer: ferramenta para projeto estruturado de sistemas. Cria e verifica um projeto da arquitetura usando projeto estruturado de sistemas. Consegue realizar transformação automática de diagramas de fluxo de dados para modelos iniciais de projeto de software. Permite ainda a construção da especificação através de processos de engenharia reversa. Está disponível para equipamentos VAX/VMS, incluindo estações de trabalho VAX e Apollo. Foi desenvolvida por *Mentor Graphics*, 8500 SW Creekside Place, Beaverton, OR 97005;

10) ER-Designer: utilizada para especificação, manutenção e prototipação de sistemas através de diagramas de Entidades-Relacionamentos. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Chen and Associates Inc.*, 4664 Constitution Ave., Ste. 1E, Baton Rouge, LA 70808;

11) Excelerator: utilizada para construção e manutenção da especificação construída através de métodos estruturados. Está disponível para equipamentos PC compatíveis sobre Microsoft Windows. Foi desenvolvida por *Index Technologies Inc.*;

12) IDMS/Architect: utilizada para especificação, manutenção e prototipação de sistemas através dos conceitos de diagramas de entidade-relacionamento, diagramas de fluxo de dados, estruturação lógica dos dados, e máquina de estados finitos. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Cullinet Software*, 400 Blue Hill Dr., Westwood, MA 02090;

13) MacBubbles: utilizada para desenvolvimento de sistemas através do método de análise estruturada proposto por Yourdon/DeMarco. Está disponível para equipamentos Macintosh. Foi desenvolvida por *StarSys INc.*, 11113 Norlee Dr., Silver Spring, MD 20902-3619;

14) System Architect: utilizada para análise, desenvolvimento e manutenção de sistemas através de técnicas estruturadas. Está disponível para equipamentos PC compatíveis sob Micro-

soft Windows 286 e 386. Foi desenvolvida por *Popkin Software Systems Inc,* 111 Prospect St., Ste. 505, Stamford, CT 06901;

I.2.2 Ferramentas Disponíveis no Brasil

1) PC-DFD: utilizada para a construção da especificação de sistemas através da análise estruturada de sistemas. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Base Tecnologia;* Rio de Janeiro, RJ;

2) PC-CASE: conjunto de ferramentas que permitem a construção e manutenção da especificação de um sistema através da utilização de técnicas estruturadas. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Base Tecnologia;* Rio de Janeiro, RJ;

3) MOSAICO: utilizada para construção de gráficos de estruturas através do projeto estruturado de sistemas. Possibilita a linearização de código para uma dada linguagem de programação a partir dos gráficos gerados. Está disponível para equipamemtos PC compatíveis. Foi desenvolvida por *IESA-TS* , Rio de Janeiro, RJ;

3) MOSAICO-DFD: utilizada para a construção de diagramas de fluxo de dados conforme método da análise estruturada de sistemas. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *IESA-TS;* Rio de Janeiro, RJ;

4) PRO-JET: utilizada para a construção de gráficos de estruturas através do projeto estruturado de sistemas. Possibilita a linearização do código para uma dada linguagem de programação a partir dos gráficos de estruturas gerados. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Wyse Informática;* Rio de Janeiro, RJ;

5) TALISMAN: utilizada para a construção do diagrama de fluxo de dados de um sistema acoplado com um dicionário de dados. Permite a definição do prório dicionário e de regras de consistência. Está disponível para equipamentos PC compatíveis. Foi desenvolvida por *Stae Informática*, Rio de Janeiro, RJ;

6) SIPS: utilizada para construção e manuteção da especificação através da análise estruturada de sistemas. Está disponível para equipamentos PC compatíveis. Foi desenvolvida pelo *CTI - Centro Tecnológico de Informática*, Campinas, SP.

I.2.3 Ferramentas Desenvolvidas na COPPE/UFRJ

1) EDIT-DFD: utilizada para desenhar diagramas de fluxo de dados. Possibilita a criação de diagramas de fluxo de dados e sua impressão. Adota a nomenclatura de análise estruturada proposta por GANE [14]. Está disponível para equipamentos PC compatíveis.AGUIAR [15];

2) DD: utilizada para construção do dicionário de dados do sistema conforme proposto por GANE [14]. Possibilita a criação, edição e impressão do dicionário de dados. Está disponível para equipamentos PC compatíveis. BLASCHEK [19];

3) EDIT-GE: utilizada para a construção dos gráficos de estrutura do projeto do sistema a partir das convenções utilizadas no projeto estruturado conforme definido por PAGE-JONES [24]. Permite a criação, edição e impressão dos diagramas. Está disponível para equipamento PC compatível. NOGUEIRA [18];

4) DARTS: utilizada para a construção e manutenção da especificação de sistemas de tempo real a partir do metódo DARTS. Possibilita a criação, edição e impressão de DFD's

generalizados e diagramas de tarefas. Está disponível para equipamento PC compatível. CAVALCANTE [22];

5) DDTR: utilizada para a construção do dicionário de dados para sistemas de tempo real. É uma extensão a ferramenta DD para análise estruturada. Está disponível para equipamentos PC compatíveis. ULLOA [23];

6) PRODOC: utilizada para a elaboração do plano de documentação para o desenvolvimento. Está disponível para equipamentos PC compatíveis. VALLE [20];

7) ESTIME: utilizada para realizar previsões de custos de desenvolvimento de produtos de software atráves de uma base de dados histórica e dos métodos COCOMO e Putman. Está disponível para equipamentos PC compatíveis. MENEZES [21]

I.3 FEGRES: FErramentas GRáficas para Engenharia de Software

Apesar das ferramentas desenvolvidas na COPPE/UFRJ suportarem uma série de atividades dentro do ciclo de desenvolvimento elas ainda não tinham facilidades que permitissem sua utilização de maneira única e integrada. O presente trabalho descreve o desenvolvimento do ambiente FEGRES - FErramentas GRáficas para Engenharia de Software.

FEGRES baseia-se no ciclo de vida em fases preocupando-se, nesta primeira fase, com a fase de especificação do sistema. Sua construção foi realizada a partir das ferramentas anteriormente desenvolvidas por AGUIAR [15] e BLASCHEK [19], acrescentando-se novas características como o controle das informações geradas a partir de um sistema gerenciador de base de dados baseado no modelo relacional , a inclusão de um editor de Diagramas de Entidades-Relacionamentos, CHEN [11], e a utilização de um sistema de interface com o usuário, baseado no Windows 2.0 desenvolvido pela MICROSOFT [25]. Uma visão simplificada de FEGRES pode

ser vista na figura (I.1). FEGRES foi modelado baseando-se nos métodos da Análise Estruturada GANE [14], Diagramas de Entidades-Relacionamentos CHEN [11] . Os Diagramas de Acesso Imediato a Dados (DAID's) segundo o método proposto por GANE [14] foram substituídos por Diagramas de Entidades-Relacionamentos CHEN [11], visando facilitar o trabalho do Engenheiro de Software.

Inicialmente estão implementados em FEGRES o editor de Diagrama de Fluxo de Dados (DFD), o editor de Diagramas de Entidades-Relacionamentos (DER) e o Dicionário de Dados (DDADOS) estendido para suportar também a descrição dos objetos contidos nos diagramas de Entidades-Relacionamentos. A principal vantagem de FEGRES em relação as ferramentas anteriormente desenvolvidas na COPPE é que este é um ambiente integrado, possuindo recursos avançados de interface e armazenamento, permitindo com isso que se obtenha uma padronização de interface com o usuário e a possibilidade de incorporação a seu núcleo de novas ferramentas que complementem o produto.

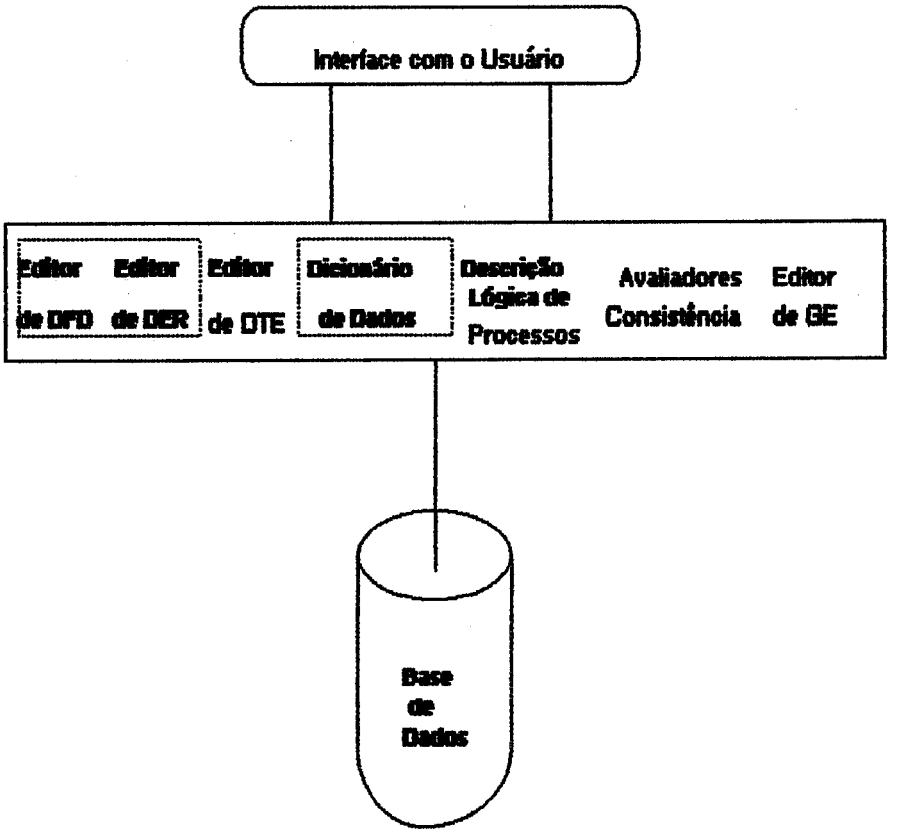

Figura I.1 - Visão Simplificada do ambiente FEGRES

I.4 Organização da Tese

No capítulo II são apresentadas as origens da análise de sistemas e é discutida uma forma de se amenizar o trabalho de especificação através da quebra do desenvolvimento através de um ciclo de vida. São apresentadas as ferramentas existentes no método da Análise Estruturada, bem como extensões propostas ao método. No capítulo III é discutida a especificação e construção de FEGRES juntamente com as dificuldades encontradas na sua construção. Características adicionais de ambientes computacionais, requisitos de interface e sistemas de armazenamento de dados capazes de suportar uma ferramenta como FEGRES também são apresentados. Conclusões e perspectivas futuras são apresentadas no final. O apêndice A contém o manual do usuário de FEGRES.

CAPÍTULO II

A ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE USANDO ANÁLISE ESTRUTURADA

II.1 A Fase de Análise

Segundo YOURDON [26] a fase de análise serve para definir e documentar os requisitos do usuário. McMENANIM [27] afirma que o propósito da análise de sistemas é produzir a essência do sistema, ou seja, definir seus verdadeiros requisitos. Um verdadeiro requisito é aquele que o produto de software deve possuir não importando que tecnologia é utilizada para construí-lo.

O *analista de sistemas é o responsável pelo levantamento dos verdadeiros requisitos* do sistema, ele é o elo de ligação entre o usuário e a equipe de desenvolvimento e sempre é responsabilizado quando o sistema não satisfaz aos requisitos do usuário. DEMARCO [28]

É preciso então evitar falhas no desenvolvimento de produtos de software através da adoção de métodos capazes de facilitar a determinação e visualização dos verdadeiros requisitos do usuário. A fase de Análise, ou Especificação, é aquela em que ocorre a determinação do que o produto deve fazer. É identificado o problema a ser resolvido e suas necessidades de interface com o hardware, outros softwares e usuário, seus requisitos funcionais e de qualidade.

A qualidade de um produto está diretamente relacionada com a qualidade de sua especificação. Quanto melhor for a qualidade da especificação menor a probabilidade de ocorrerem necesidades de correções em erros existentes por mal levantamento dos dados nas fases posteriores do ciclo de desenvolvimento, consequentemente, uma redução dos custos e tempo de desenvolvimento é obtida. Uma especificação de boa qualidade, que retrate fielmente

as necessidades do usuário, pode ser obtida a partir da utilização de métodos apropriados.

ROCHA [29] diz que a qualidade da especificação é determinada através de três objetivos utilizabilidade, confiabilidade conceitual e confiabilidade da representação. O objetivo utilizabilidade diz respeito a utilização da especificação. Uma especificação existe para ser usada. Mas para que uma especificação possa ser utilizada é necessário, também, que ela retrate fielmente as necessidades e expectativas do usuário, atingindo com isto a confiabilidade conceitual, e possua uma apresentação adequada ao bom entendimento por parte dos que a utilizam, atingindo a confiabilidade da representação. Se os documentos produzidos na especificação não conseguem apresentar claramente seu conteúdo e são de difícil manipulação, torna-se muito difícil se utilizar a especificação.

A construção de uma especificação de boa qualidade é o principal fator de sucesso num projeto. É através dela que o usuário poderá avaliar se o que está sendo documentado realmente retrata suas necessidades e expectativas a respeito do produto. Especificações construídas através de métodos que não proporcionam uma comunicação adequada com o usuário, utilizando-se de conceitos e técnicas de difícil acesso por parte de pessoas não qualificadas, tornam o processo de desenvolvimento uma tarefa lenta e cheia de riscos. A utilização destes tipos de métodos podem provocar a não detecção de erros em fases iniciais do projeto, provocando com isso um aumento do custo do produto, e o que é pior, a não identificação se o que está sendo construído realmente representa o que se deseja construir.

Especificações podem ser construídas sobre três enfoques, BOEHM [30]. Podemos ter especificações escritas de maneira informal através do uso de linguagem natural. Este tipo de especificação pode ser construída por qualquer indivíduo pois não requer qualquer treinamento específico para seu uso. Em compensação a quantidade de mal-entendidos entre usuários e desenvolvedores tende a ser alta devido às ambiguidades que a linguagem natural possui. Um outro enfoque que pode ser utilizado é o de se construirem especificações em linguagens semi-formais, ou formatadas. Estas linguagens proporcionam uma sintaxe padronizada para se

construir as especificações, reduzindo em muito as ambiguidades. Necessitam de um nível médio de treinamento para aprendizado e leitura. Por fim podemos construir especificações através de métodos formais. Este enfoque consegue produzir especificações sem ambiguidades devido a utilização de uma forma matemática precisa. Exigem um alto grau de treinamento para leitura e aprendizado do método.

A adoção de um ou outro enfoque deve ser realizado de acordo com o tipo de sistema que se deseja construir. Se estamos especificando um sistema administrativo é mais interessante que utilizemos um enfoque semi-formal do que um formal. Mas se estamos especificando um sistema de acompanhamento médico para ser utilizado em pacientes internados em centros cirúrgicos a especificação feita de maneira formal é mais segura e confiável.

II.1.1 ANÁLISE ESTRUTURADA

Na década de setenta a preocupação em se obter especificações de requisitos que representassem de uma forma clara e precisa as necessidades do usuário, proporcionou a elaboração de vários estudos nesta área.

Estes estudos levaram à publicação em 1975, por ROSS e SCHOMAN [13], do que poderia ser chamado de embrião da Análise Estruturada. De acordo com MCMENAMIN [27] este foi o maior passo na tecnologia da definição de requisitos, pois propunha a adoção de um conjunto de ferramentas gráficas de modelagem que eliminavam a maioria das deficiências encontradas nas especificações narrativas. Uma grande deficiência deste trabalho, entretanto, foi não dizer muito a respeito do processo de desenvolvimento de uma especificação de requisitos.

A partir deste estudo preliminar, DEMARCO [28], YOURDON [26] e GANE [14] propuseram versões semelhantes de Análise Estruturada. Estas propostas sugeriem que a Análise Estruturada fosse feita através das seguintes ferramentas, ligeiramente modificadas entre cada proposta:

1. Diagrama de Fluxo de Dados
2. Dicionário de Dados
3. Descrição da Lógica dos Processos
4. Diagramas de Acesso Imediato a Dados

De acordo com YOURDON [17] Análise Estruturada é o uso de ferramentas gráficas de documentação para produzir um novo tipo de especificação funcional, ou seja, uma especificação estruturada.

A evolução no desenvolvimento dos produtos de software, juntamente com a necessidade de se representar novas características nestes produtos , fez com que algumas modificações fossem introduzidas nas propostas originais, e atualmente tem-se as seguintes ferramentas para a construção de uma especificação estruturada, segundo YOURDON [17]:

1. Diagramas de Fluxo de Dados;
2. Dicionário de Dados;
3. Descrição da Lógica dos Processos;
4. Diagramas de Entidades-Relacionamentos;
5. Diagramas de Transição de Estados.

Estas sugestões visam dar maior flexibilidade e ao mesmo tempo aumentar a produtividade da equipe de desenvolvimento. A possibilidade de visualizar o produto que está sendo especificado sob vários pontos de vista melhora a comunicação com o usuário, e junto com isto, possibilita um processo de verificação e validação do que está sendo construído mais rápido e seguro.

II.2 Diagrama de Fluxo de Dados

II.2.1 Descrição Geral

Os diagramas de fluxo de dados permitem uma fácil visualização dos fluxos de informação através do sistema, seja ele manual, automatizado ou uma mistura de ambos.

A construção de um diagrama de fluxo de dados pode ser realizada através de uma abordagem "Top-Down". Um sistema típico pode ser representado através de várias expansões de um diagrama de fluxo de dados. Estas expansões são conhecidas como "explosões". O primeiro nível, nível 0 (zero) ou diagrama de contexto, representa as interfaces do sistema com o ambiente externo, não se preocupando em apresentar a lógica interna do sistema. No segundo nível, ou nível 1, são representados, além das interfaces do sistema com o ambiente externo, as atividades essenciais do sistema juntamente com os depósitos de dados principais. A partir daí torna-se necessária a expansão do diagrama de fluxo de dados até que seja possível identificar todos os detalhes do sistema, permitindo assim uma assimilação do todo que o mesmo representa. Essa expansão é feita a partir da explosão dos processos, que nos permite chegar a um nível de detalhamento adequado. Na figura (II.1) temos a representação do que seria uma explosão de um processo.

II.2.2 Elementos de um DFD

GANE [14] sugere que um diagrama de fluxo de dados seja representado por quatro elementos principais que são o Processo, o Depósito de Dados, a Entidade Externa, e o Fluxo de Dados. A figura (II.2) mostra a representação gráfica destes elementos.

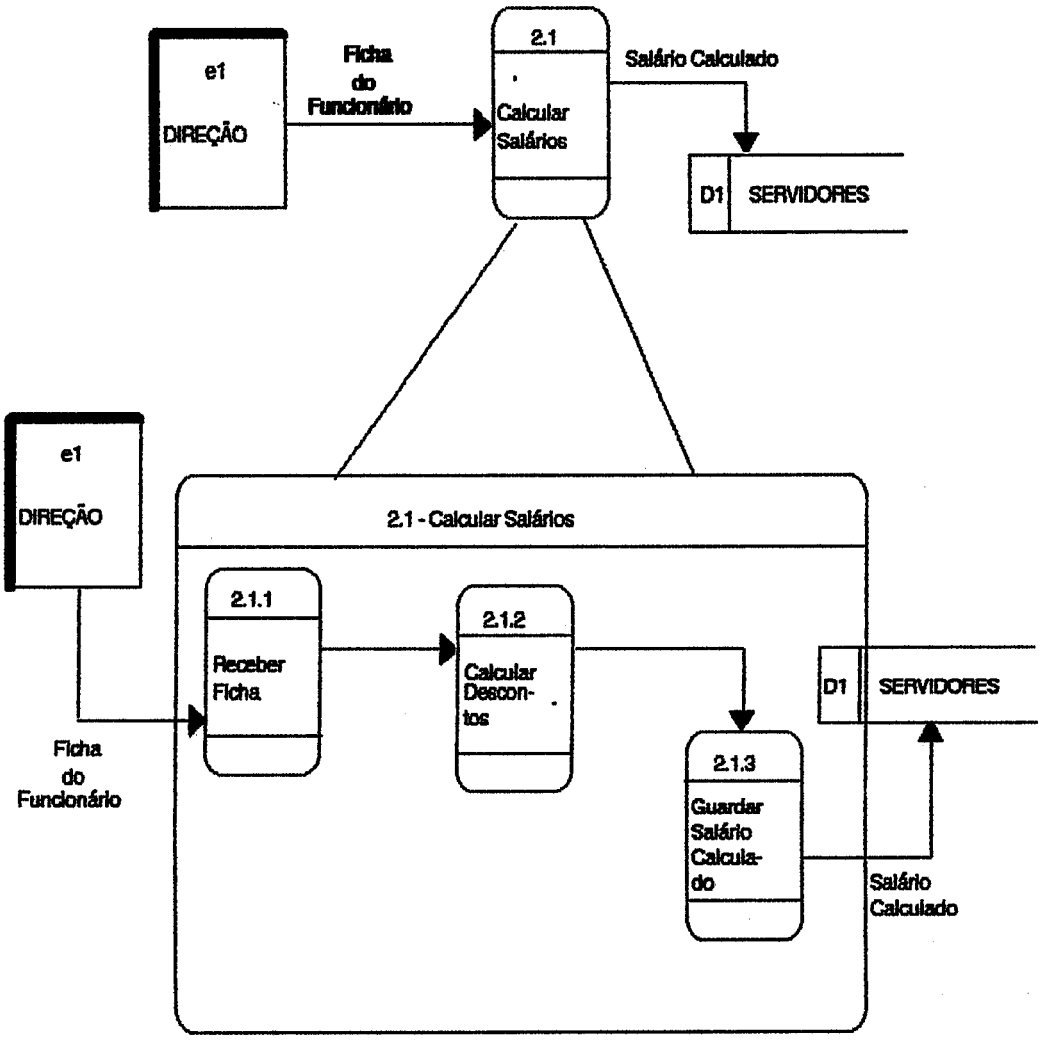

Figura II.1 - Um DFD com a explosão de um Processo

Uma Entidade Externa é qualquer entidade lógica (por exemplo pessoas, organizações, outros sistemas) que não faz parte dos limites do sistema mas que fornece ou recebe dados do sistema.

Pode ocorrer a necessidade de repetição ou duplicação da entidade externa no diagrama de fluxo de dados. Quando esta situação ocorre, marcas são colocadas na canto inferior direito conforme o número de entidades externas repetidas. A representação de entidades externas repetidas pode ser vista na figura (II.3).

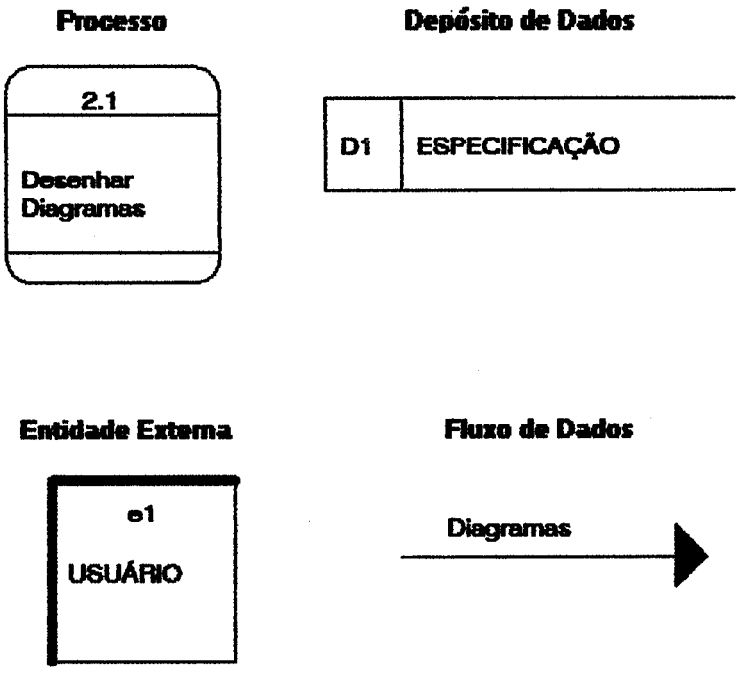

Figura II.2 - Representação Gráfica dos Elementos do DFD

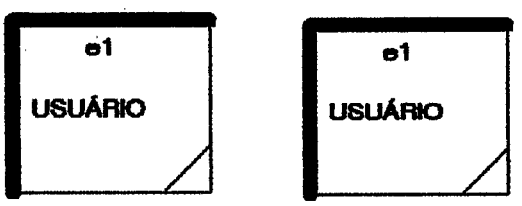

Figura II.3 - Representação de Entidades Externas Repetidas

Enquanto as entidades externas são as responsáveis em trocar informações com o sistema, os Processos são os responsáveis pela transformação dessas informações.

Em sua representação gráfica encontramos três áreas distintas:

- identificação no DFD
- identificação funcional
- identificação do responsável pela função.

Algumas características devem ser ressaltadas sobre os processos:

.a identificação funcional se faz através de uma sentença possuindo um verbo no imperativo identificando a função;

.um processo, através de uma abordagem "Top-Down", pode ser representado por outros processos. Esta subdivisão implica numa referência ao processo pai em sua identificação no DFD.

Na figura (II.4) podemos ver a representação de um processo.

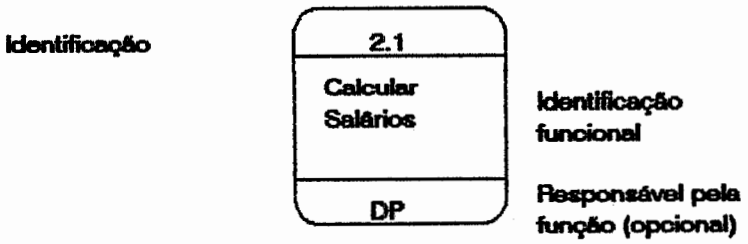

Figura II.4 - Representação de um Processo no DFD

Os Depósitos de Dados são os responsáveis pelo armazenamento dos dados existentes no sistema e que necessitam ser guardados.

Como ocorre com as entidades externas, a repetição ou duplicação dos depósitos de dados num diagrama de fluxo de dados é feita colocando-se marcas de repetição no lado esquerdo de sua representação. A figura (II.5) mostra como deve ser feita esta representação.

Figura II.5 - Representação de Depósitos de Dados Repetidos num DFD

Os Fluxos de Dados são os responsáveis pelo transporte dos dados entre processos, processos e depósitos de dados e processos e entidade externas. Fluxos de dados podem transportar estruturas de dados e/ou elementos de dados.

Elementos de Dados são definidos como itens elementares, ou seja, aqueles que não suportam uma subdivisão.

Uma Estrutura de Dados, por sua vez, é definida como um item de dado composto cuja representação pode ser feita a partir de outras estruturas de dados e de elementos de dados.

Um fluxo de dados precisa ter uma origem e um destino, cujo sentido é indicado pela direção da seta. Na figura (II.6) temos as possíveis representações de um fluxo de dados.

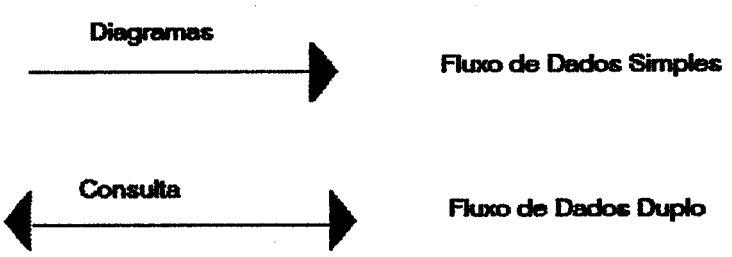

Figura II.6 - Representação de Fluxo de Dados no DFD

II.3 Dicionário de Dados

II.3.1 Descrição Geral

Um dicionário de dados é uma coleção organizada das definições lógicas de todos os nomes de dados que aparecem no diagrama de fluxo de dados, YOURDON [17]

A necessidade de possuirmos um dicionário de dados surge de uma maneira natural tendo em vista a quantidade de informação que aparece na especificação de um produto e que necessita ser descrita.

Sua utilização pode ser feita de forma manual ou automatizada. O uso do Dicionário de Dados de forma manual é extremamente trabalhoso e ineficiente devido ao grande volume de informação que precisa ser documentada na construção da especificação de um produto de software.

II.3.2 Elementos Descritos

Cada tipo de elemento existente no Dicionário de Dados possui um modelo específico de ficha para documentação.

Os elementos existentes num Dicionário de Dados surgem durante a construção da especificação utilizando-se a Análise Estruturada. Fichas para processos, depósitos, entidades externas, fluxo de dados, estruturas de dados, elementos de dados e itens de glossário são armazenadas através dele. Estas fichas foram desenhadas de forma a simplificar ao máximo o trabalho de armazenamento e manuseio das mesmas por parte do projetista.

Nas figuras (II.7) a (II.13) podemos observar como são as fichas para cada tipo de elemento, conforme BLASCHEK [19].

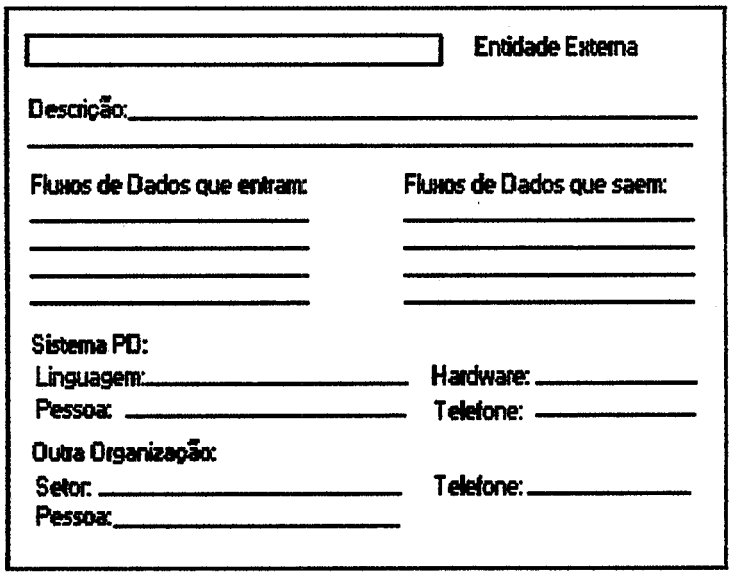

Figura II.7 - Ficha para Descrição de Entidade Externa

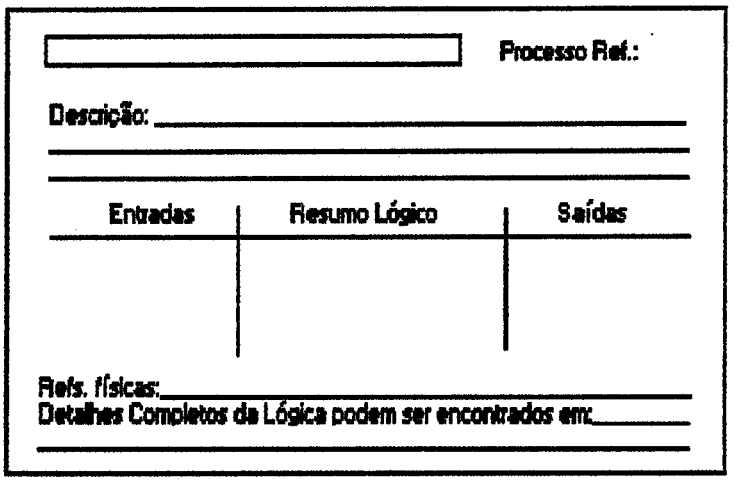

Figura II.8 - Ficha para Descrição de Processo

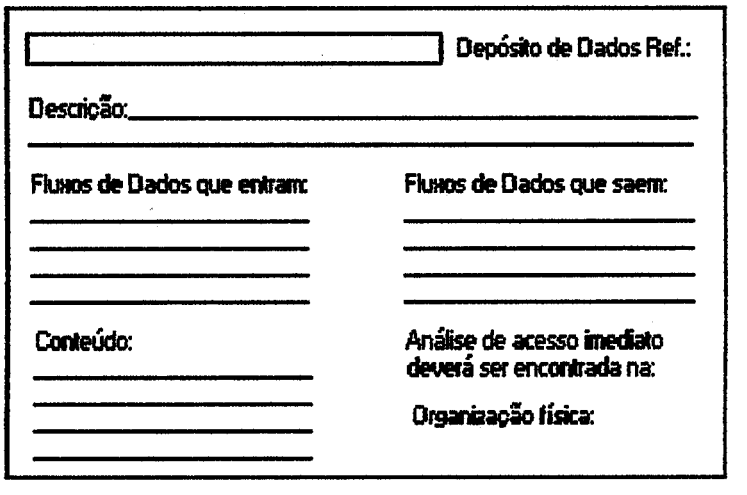

Figura II.9 - Ficha para Descrição de Depósito de Dados

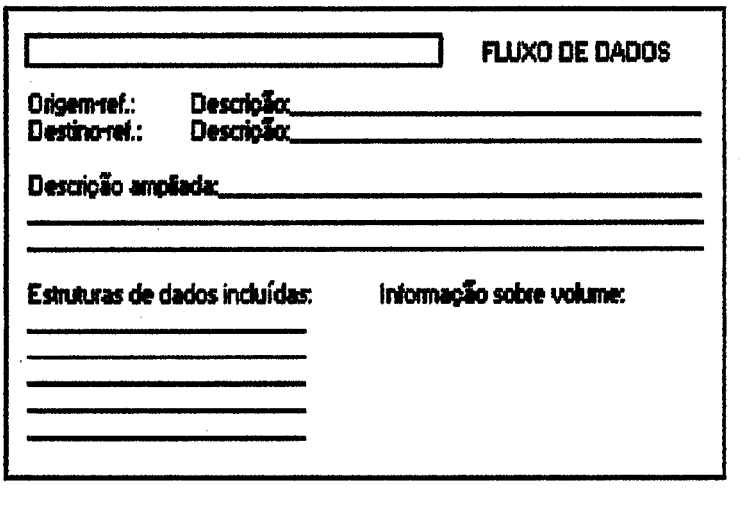

Figura II.10 - Ficha para Descrição de Fluxo de Dados

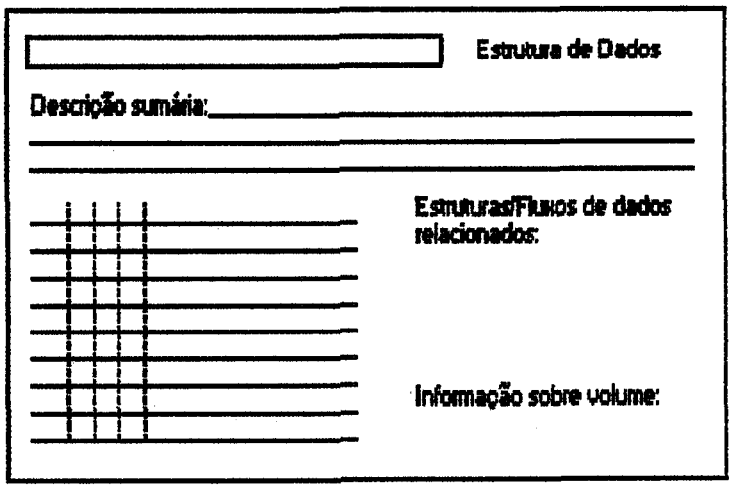

Figura II.11 - Ficha para Descrição de Estruturas de Dados

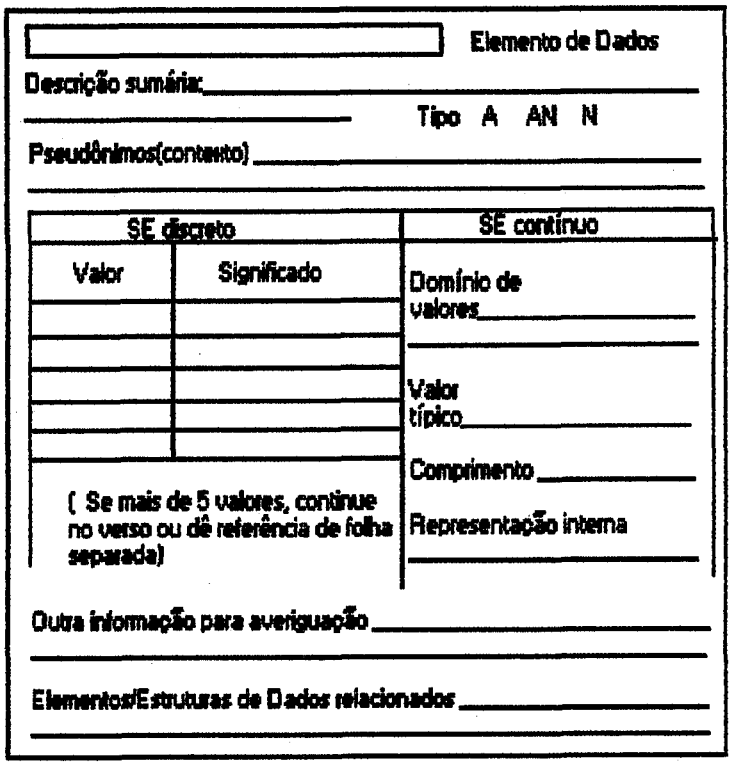

Figura II.12 - Ficha para Descrição de Elemento de Dados

Um estudo mais detalhado sobre Dicionário de Dados pode ser encontrado em BLASCHEK [19] e ULLOA [23].

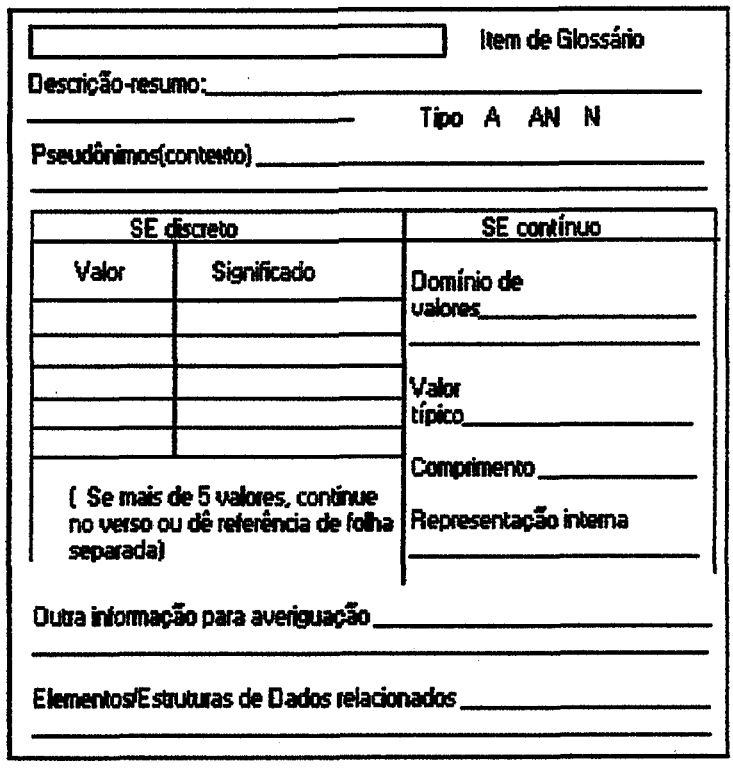

Figura II.13 - Ficha para Descrição de Item de Glossário

A utilização de um Dicionário de Dados pode ser realizada de forma manual ou automatizada, sendo que esta última facilita em muito o trabalho do projetista da aplicação quando o mesmo necessita consultar as fichas dos elementos. Podemos destacar as seguintes vantagens do Dicionário de Dados automatizado em comparação com o manual:

1) Armazenamento das informações em uma base de dados única, acessível por toda a equipe de desenvolvimento,

2) Possibilidade de cruzar informações de forma eficiente, facilitando a obtenção de

listagens específicas,

3) Diminuição do tempo para obtenção de informações,

4) Possibilidade de utilizar um sistema gerenciador de base de dados para controlar os acessos, aumentando ainda mais a eficiência e manutenção da consistência das informações.

II.4 Descrição da Lógica dos Processos

II.4.1 Descrição Geral

Apenas os Diagramas de Fluxo de Dados e o Dicionário de Dados não conseguem representar detalhadamente a lógica dos processos existentes no sistema. A Descrição da Lógica do Processo necessita ser realizada através de uma linguagem mais clara e sem as ambiguidades de nossa linguagem corrente, que é inadequada para tal fim devido as armadilhas sintáticas e semânticas que a mesma possui.

Segundo GANE[14] os processos podem ser descritos através de quatro linguagens, ou ferramentas, distintas que podem ser usadas em conjunto ou separadamente, chamadas Tabela de Decisão, Árvore de Decisão, Português Estruturado/Pseudocódigo e Português Compacto.

Tabelas de Decisão são utilizadas para representar a lógica de processos que possuam um grande número de ações juntamente com um grande conjunto de combinações de condições entre estas ações.

Árvores de Decisão são utilizadas para representar a lógica de processos que possuam um pequeno número de ações juntamente com poucas combinações de condições.

Português Estruturado/Pseudocódigo e Português Compacto se utilizam de uma forma de representação narrativa da lógica do processo. São adequadas para representar processos

que sejam mais simples e não envolvam uma lógica com estruturas de decisão complexa em sua descrição.

II.5 Diagramas de Entidades-Relacionamentos

II.5.1 Descrição Geral

Diagramas de Entidades-Relacionamentos foram inicialmente propostos por CHEN [11] para representação do modelo de dados de um sistema.

O propósito de um Diagrama de Entidades-Relacionamentos (DER), segundo YOURDON [17] , é mostrar os principais objetos de dados tratados pelo sistema e o relacionamento entre eles.

Além de possuir uma forma de representação simples e poderosa, diagramas de Entidades-Relacionamentos amenizam o processo de aprendizado por parte da equipe de desenvolvimento devido a abordagem "Top-Down" utilizada na construção do modelo completo de Entidades-Relacionamentos, abordagem esta comum à equipe de desenvolvimento e largamente utilizada na construção de diagramas de fluxo de dados, CHRISMAN [31].

II.5.2 Elementos de um DER

Os elementos básicos para representação de um Diagrama de Entidades-Relacionamentos são a Entidade e o Relacionamento.

Em sua definição inicial, CHEN [11] determina que uma Entidade é uma coisa que pode ser identificada distintamente. Autores que realizaram estudos mais recentes, de uma maneira geral, definem uma Entidade como sendo uma coisa (objeto, conceito) a qual a empresa reconhece como sendo capaz de existir independentemente e que pode ser identificada de maneira única. HOWE [32] ,JACKSON [33], SETZER[34]. A única característica de definição que pode ajudar a encontrar o que é entidade num sistema é que entidade usualmente é um nome.

Um Relacionamento é uma associação ou conexão, entre duas ou mais entidades, JACKSON[33], HOWE[32].

Na figura (II.14) temos a representação de Entidades e Relacionamentos. Na figura (II.15) representamos a associação entre entidades através de seu relacionamento. A principal forma de se encontrar relacionamentos é notar que um relacionamento é normalmente um verbo.

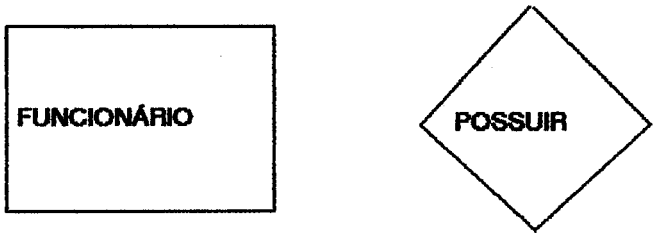

Figura II.14- Representação de Entidade e Relacionamento

Assim temos que entidades podem ser representadas por ALUNO, PROFESSOR, DISCIPLINA, BANCO, CLIENTE, etc. e alguns relacionamentos entre estas entidades seriam PROFESSOR LECIONA DISCIPLINA, BANCO POSSUI CLIENTES, ALUNO CURSA DISCIPLINA, dentre outros.

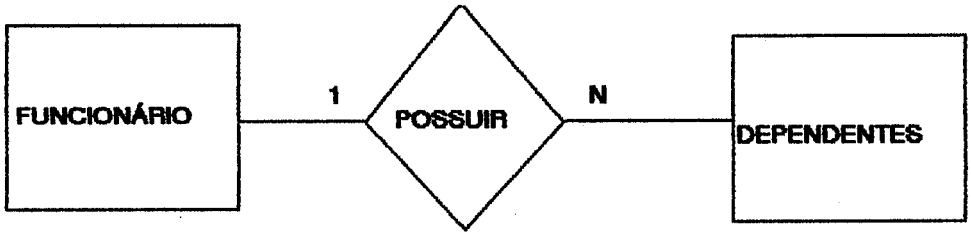

Figura II.15 - Diagrama de Entidades-Relacionamentos

Podemos encontrar três formas básicas de Relacionamentos:

. 1:1 - quando uma entidade se relaciona com apenas uma outra entidade

. 1:N - quando uma entidade se relaciona com várias outras entidades

. M:N - quando uma entidade se relaciona com várias outras entidades em ambos os sentidos.

Um Relacionamento é dito binário quando apenas duas entidades participam do relacionamento. É chamado ternário quando três Entidades participam do relacionamento. O sentido em que visualizamos o relacionamento pode identificar o papel, CHEN [11], de uma entidade no relacionamento.

Além dos conceitos de Entidade e Relacionamento temos que levar em consideração também o conceito de Atributo. Um Atributo é uma propriedade de uma entidade, JACKSON [33], HOWE [32].

Com os atributos podemos destacar características específicas das entidades. Alguns destes atributos, devido a propriedades próprias, conseguem identificar uma Entidade. Neste caso eles são chamados de Atributos Dominantes. Não são apenas as Entidades que possuem Atributos. Muitas vezes necessitamos representar Relacionamentos que possuem atributos. Isto geralmente ocorre quando a cardinalidade do relacionamento é M:N. Neste caso os atributos são

conhecidos como atributos do relacionamento.

Caso haja a necessidade de decompormos relacionamentos do tipo M:N para 1:N podemos utilizar as entidades compostas. Esta decomposição irá evitar que relacionamentos possuam atributos, simplificando o modelo.

II.5.3 Extensões Utilizadas

O Modelo de Entidade-Relacionamento na sua forma mais pura, CHEN [11], não consegue representar de maneira precisa os vários relacionamentos existentes entre as entidades. O simples fato de surgir um relacionamento entre uma entidade e um conjunto de entidades que possuam características em comum inviabilizam seu uso. SMITH [35] definiu duas extensões ao modelo que são a agregação e a generalização.

Uma agregação refere-se a uma abstração na qual o relacionamento entre os objetos é considerado como um objeto de mais alto nível. Neste caso poderíamos ter que o relacionamento entre as entidades Funcionário, Esposa e Filhos poderia ser visto como uma agregação denominada Família.

Uma generalização é a reunião de objetos similares provocando o aparecimento de um objeto genérico. Um caso típico é quando encontramos as várias classes de empregados de uma empresa, que podem ser técnicos, engenheiros, químicos, etc. que denominamos Funcionários.

II.6 Diagramas de Transição de Estados

II.6.1 Descrição Geral

O uso de Diagramas de Transição de Estados para especificação e projeto de diálogos foi primeiro proposto por PARNAS [36] e tem sido usado para especificar requisitos de produtos de software por CASEY [37].

Os Diagramas de Transição de Estados (DTE) representam a dinâmica do sistema sobre o tempo. Devem ser utilizados quando a variação dos possíveis estados que o sistema pode assumir for importante para sua compreensão.

Atualmente têm se utilizado Diagramas de Transição de Estados para a modelagem de protótipos de interface com o usuário para um determinado sistema HEKMATPOUR [38], e são também extremamente utéis para a especificação de protótipos de esquemas de ajuda ("help") dentro de sistemas. Um Diagrama de Transição de Estados pode ser visualizado como sendo um gráfico orientado consistindo de nós, representados por um círculo, e caminhos, representados por ligações entre esses nós, que mostram os estados e eventos do sistema respectivamente.

O DTE tem como função, ULLOA [23]:

.mostrar as origens e os destinos das transições que são ocasionadas por eventos;
.identificar todas as condições de transferência de controle.

II.6.2 Elementos de um DTE

São os seguintes os elementos utilizados para representação de um DTE:

.Estado: são situações estabelecidas que definem o comportamento do sistema. é representado por um círculo no diagrama.

.Evento: provoca uma transição de estado entre um estado e outro no sistema. É representado por uma seta no diagrama.

Vale ressaltar neste ponto algumas observações entre a ocorrência de eventos e as transições correspondentes ULLOA [23]:

- nem todos os eventos necessariamente causarão uma transição a partir de um dado estado;
- um dado evento nem sempre causa uma transição dentro de um mesmo estado, e,
- mais de um evento pode causar a mesma transição de estado.

Nas figuras (II.16) e (II.17) podemos ver os elementos que compõem um Diagrama de Transição de Estados e a representação de um diagrama.

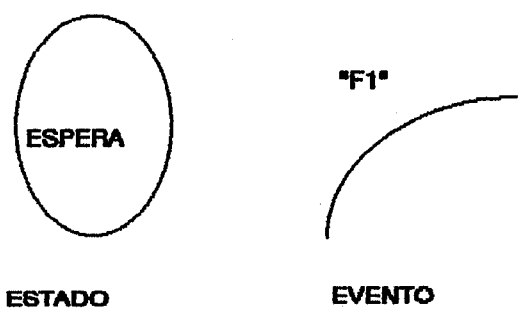

Figura II.16- Representação dos Elementos de um DTE

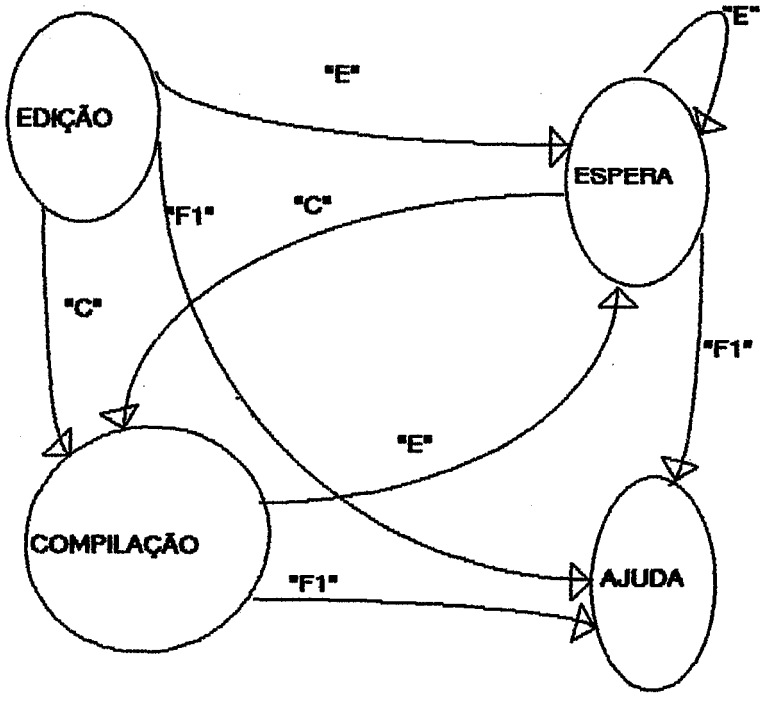

Figura II.17- Diagrama de Transição de Estados

II.7 Justificativas para extensões ao método Análise Estruturada

Análise Estruturada, conforme proposta por GANE [14], possui ferramentas próprias para a construção da especificação de um produto de software. O uso de Análise Estruturada na sua forma original não consegue representar todas as características necessárias a visualizar o produto sob o aspecto temporal e do modelo de dados, permitindo apenas que se visualize o aspecto funcional. Isto na maioria das vezes impossibilita uma validação precisa do que está sendo construído pois o usuário não consegue enxergar todo o produto de maneira clara e consistente.

Existem atualmente propostas que estendem a Análise Estruturada de maneira a acrescentar em seu contexto ferramentas que permitam modelar o produto sob o aspecto temporal e sob o aspecto do modelo de dados do sistema. YOURDON [17] propõem a substituição de diagramas de acesso imediato a dados, DAID's, usados para representar o relacionamento entre os objetos de dados do sistema, por diagramas de Entidades-Relacionamentos, e inclui, também, diagramas de Transição de Estados para a representação das variações de estado que o sistema pode sofrer.

A utilização de diagramas de Acesso Imediato a Dados se torna conveniente para pequenos sistemas onde a existência de objetos de dados e relacionamentos entre eles é pequena. Além disso, o tratamento destes objetos juntamente com o refinamento de sua representação deve ser feito através de um processo de normalização CODD [39] [40]. Este processo é extremamente trabalhoso e se torna uma tarefa árdua para o Engenheiro de Software quando o mesmo tem de lidar com sistemas que possuam um conjunto de objetos com grande volume de elementos. JACKSON [33] afirma que, apesar de ser uma opinião pessoal a quantidade de informação que torna o uso deste tipo de abordagem complicado, a maioria das pessoas diz que acima de vinte elementos o uso de tal método pode ser desastroso. Neste caso outros métodos de projeto podem ser usados para simplificar o trabalho e dentre eles se destaca o Modelo de Entidades-Relacionamentos. CHEN [11]

Uma outra característica interessante a se destacar é que DFDs e DERs representam dois aspectos diferentes do mesmo sistema, enquanto o primeiro representa o fluxo de informação e as transformações sofridas por elas no sistema, o segundo representa as informações e o relacionamento entre elas, consequentemente, uma correspondência biunívoca entre as duas representações pode ser utilizada para verificação da consistência do modelo: cada depósito de dados representado no DFD possui pelo menos um objeto correspondente no DER.

A utilização de Diagramas de Entidades-Relacionamentos facilita muito o entendimento

do modelo de dados representado por pessoas que não são da área de processamento de dados, permitindo que uma verificação mais rigorosa possa ser realizada sobre os modelo de dados.

CAPÍTULO III

FEGRES: Um Conjunto de Ferramentas Gráficas para Engenharia de Software

III.1 INTRODUÇÃO

FEGRES é um conjunto de ferramentas para Engenharia de Software. Idealizada para ser utilizada em projetos que se baseiam no ciclo de desenvolvimento em fases, sua versão inicial preocupa-se com a fase de especificação do produto. Sua construção foi realizada a partir das ferramentas anteriormente desenvolvidas por AGUIAR [15] e BLASCHEK [19], acrescentando-se novas características, como o controle das informações geradas a partir de um sistema gerenciador de base de dados baseado no modelo relacional , a inclusão de um editor de Diagramas de Entidades-Relacionamentos, CHEN [11], e a utilização de um sistema de interface com o usuário, baseado no Windows 2.0 desenvolvido pela MICROSOFT [25]. Uma visão simplificada do estágio atual de FEGRES pode ser vista na figura (III.1).

Inicialmente estão implementados em FEGRES o editor de Diagrama de Fluxo de Dados, o editor de Diagramas de Entidades-Relacionamentos e o Dicionário de Dados estendido para suportar também a descrição dos objetos contidos nos diagramas de Entidade-Relacionamento. A principal vantagem de FEGRES em relação as ferramentas anteriores é que seu conjunto de ferramentas executa num ambiente integrado possuindo recursos avançados de interface e armazenamento, permitindo com isso que se obtenha uma padronização da interface com o usuário e a possibilidade de incorporação a seu núcleo de novas ferramentas que complementem o produto.

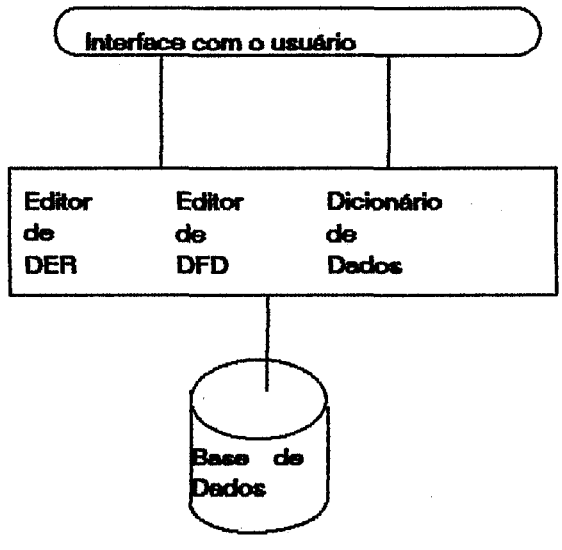

Figura III.1 - Estágio Atual de FEGRES

III.2 ESPECIFICAÇÃO DA FERRAMENTA

III.2.1 Requisitos Funcionais

FEGRES foi projetada para suportar a construção de produtos que se baseiam no ciclo de desenvolvimento por fases, conforme definido por FAIRLEY [41]. Em sua versão inicial preocupa-se com a fase de especificação do sistema. Nesta fase o analista define o que o sistema deve fazer e, para representação desta especificação, gera uma série de documentos que devem ser construídos de maneira consistente e uniforme. Para suportar a construção destes documentos e ao mesmo tempo proporcionar ao desenvolvedor da aplicação um ambiente ameno ao trabalho e de rápido aprendizado , FEGRES utiliza-se de métodos estruturados para o desenvolvimento de sistemas.

A construção de produtos de software através de métodos estruturados proporciona ao desenvolvedor da aplicação um conjunto de instrumentos apropriados e que são,de certa forma, complexos se utilizados de forma manual. A implementação destes instrumentos através de

ferramentas específicas cria um ambiente propício para o desenvolvimento de aplicações, atingindo um grau de qualidade e produtividade adequados. Dentre os métodos estruturados foi escolhido Análise Estruturada de Sistemas, GANE [14], como o método apropriado para ser suportado por FEGRES. Apesar de ser um método largamente utilizado pelos desenvolvedores de sistemas permite, ainda, que novas ferramentas sejam acrescidas visando o suporte a outras fases do desenvolvimento e se aproveitando das informações geradas por ele. Visando efetivar uma melhora no desenvolvimento de produtos de software, e ao mesmo tempo, proporcionar ao desenvolvedor da aplicação instrumentos que possibilitem uma visão sobre outro ponto de vista do produto que se está construindo, FEGRES permite ao desenvolvedor da aplicação utilizar extensões ao método de Análise Estruturada, GANE [14]. Estas extensões , também definidas por YOURDON [17], visam complementar os instrumentos existentes, e aumentar a confiabilidade da especificação.

Duas extensões principais, nesta versão, são suportadas por FEGRES. Os diagramas de acesso imediato a dados, DAID's, foram substituídos por diagramas de Entidades-Relacionamentos, CHEN [11]. Os objetivos principais de se utilizar DER no lugar de DAID's são a simplificação do processo de normalização, CODD [39] [40], e a visualização da especificação do produto sob o ponto de vista dos dados, ou objetos, componentes do sistema. Um outro recurso adicional é a extensão do dicionário de dados para poder armazenar os objetos existentes no DER. Isto se faz necessário devido à necessidade de se manter uma correspondência biunívoca existente entre os modelos funcional e de dados no sistema: os objetos existentes no DER de alguma forma têm que estar documentados no DFD, e os objetos armazenados nos depósitos de dados do sistema de alguma forma têm de estar relacionados no DER. Esta correspondência pode ser melhor explicada da seguinte forma: todo objeto, ou seus atributos, descrito no DER tem que estar armazenado em um ou mais depósitos de dados, e os elementos armazenados nos depósitos de dados são representados através do modelo apresentado pelo DER.

A utilização de DER possibilita, também, um meio de comunicação mais acessível pelo

usuário não conhecedor das técnicas utilizadas na modelagem de sistemas, facilitando com isto que uma verificação e validação da especificação seja realizada aumentado a garantia de o que está sendo construído realmente é o que está sendo pedido. Nas figuras (III.2) a (III.8) podemos ver os diagramas de fluxo de dados que descrevem FEGRES. Foram gerados utilizando o EDIT-DFD, AGUIAR [15].

III.2.2 Requisitos de Interface

A interação com o computador é um processo em dois sentidos: num o computador apresenta a informação para o usuário, no outro, o usuário fornece comandos e dados para o computador, NEWMAN [42]. Inicialmente, quando surgiram os primeiros equipamentos, esta interação era feita através de chaves nos painéis, os equipamentos eram precários, apesar de representarem um grande avanço para a época, e não permitiam uma comunicação de forma simples e clara. Com o passar dos tempos o grande avanço tecnológico foi proporcionando o aparecimento de equipamentos computacionais mais potentes, passamos pelos cartões perfurados, e atualmente, equipamentos computacionais cada vez mais poderosos, capazes de gerar imagens de alta resolução aliadas com o controle interativo sobre elas, estão se constituindo num novo meio de apresentação de informação e conceitos. Dispositivos especiais, especificamente projetados para o processo de interação, já são de fácil acesso para o usuário , dentre os quais podemos destacar o mouse, canetas de luz, traçadores , dentre outros. PERSIANO [43]

A grande disponibilidade destes equipamentos , provocando consequentemente uma redução nos custos de instalação e aquisição por parte do usuário, tem proporcionado que cada vez mais pessoas tenham acesso a eles, permitindo com isso que um novo elemento passe a fazer parte do cotidiano como uma ferramenta de trabalho e lazer. Na maioria das vezes as pessoas que utilizam estes equipamento não são especialistas em informática, o que causa a necessidade de se definir padrões de comunicação que facilitem seu uso.

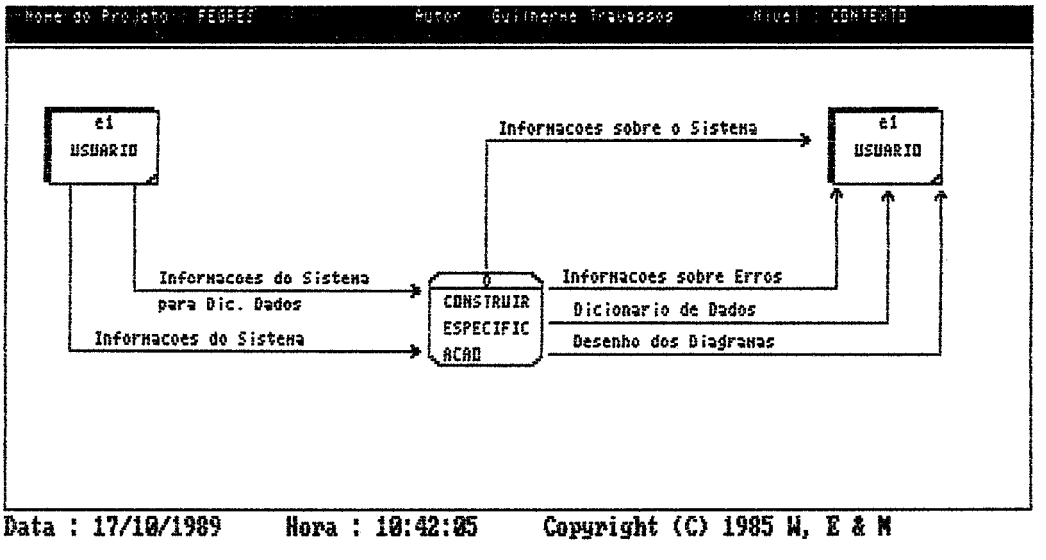

Figura III.2 - DFD nível contexto

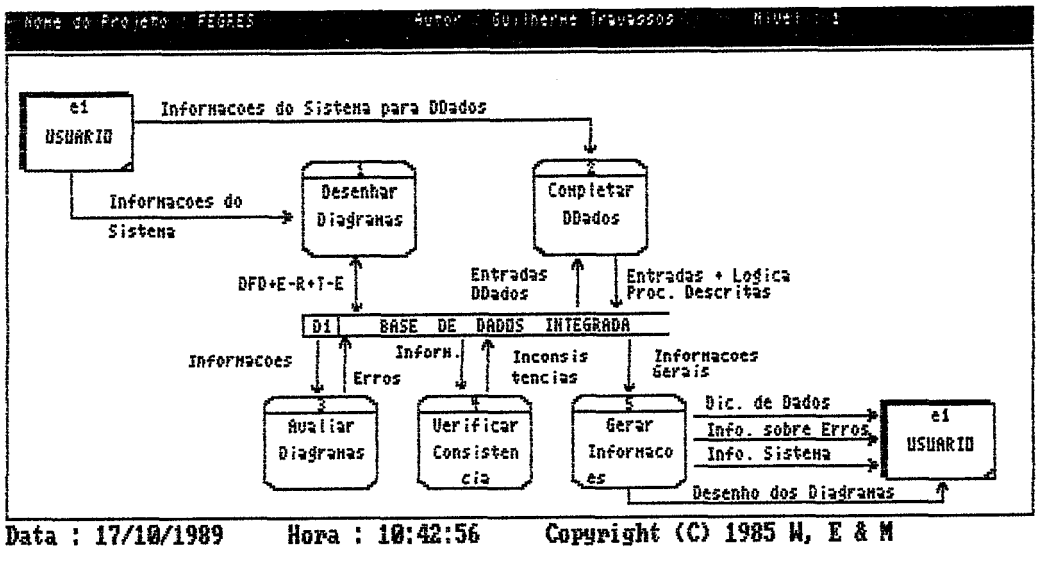

Figura III.3 - DFD nível 1

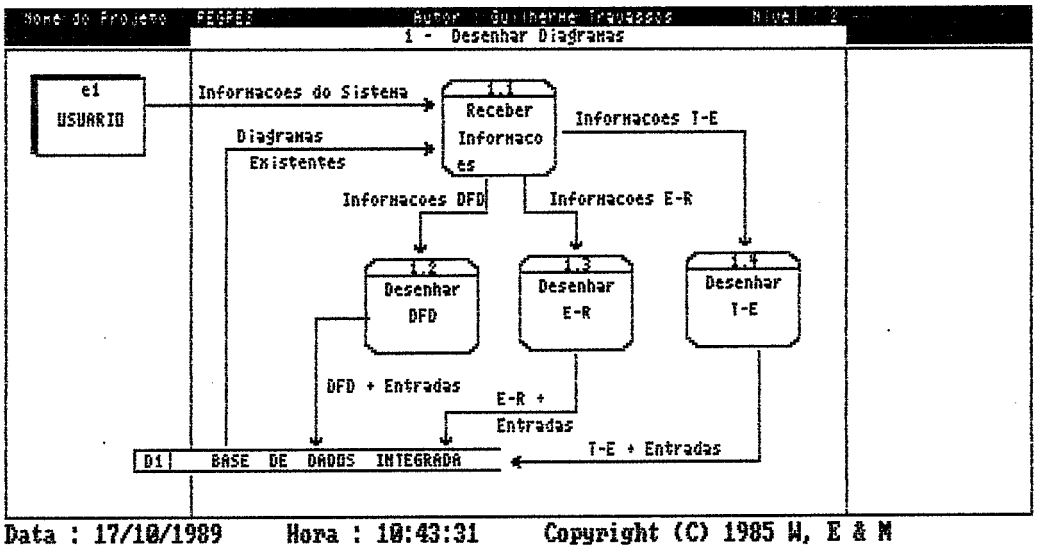

Figura III.4 - DFD explosão do processo 1

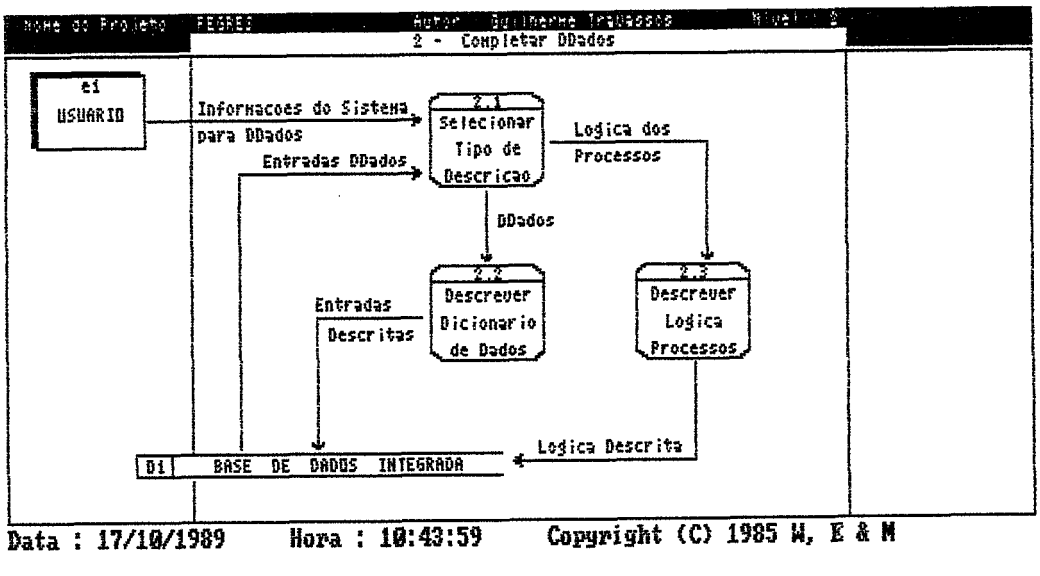

Figura III.5 - DFD explosão do processo 2

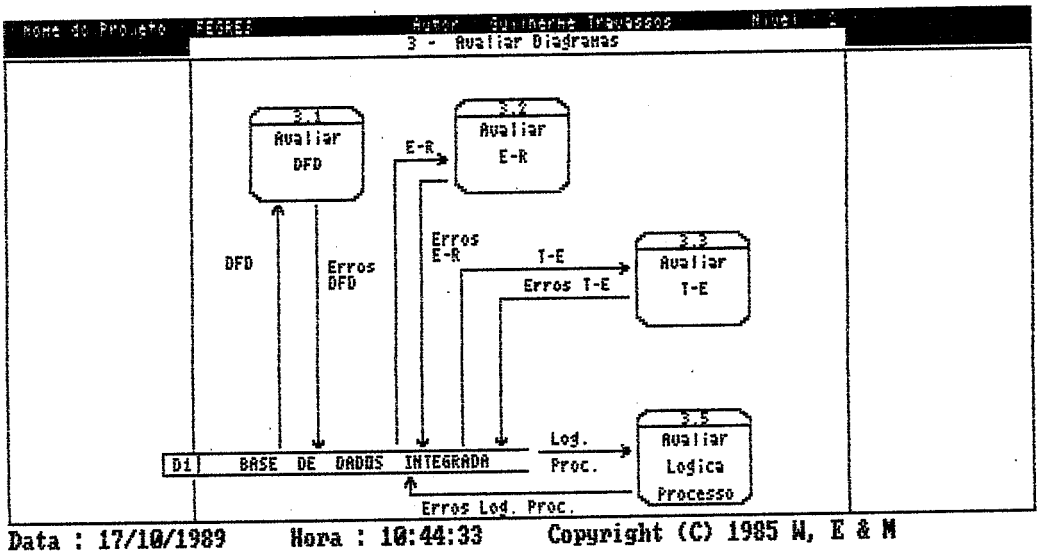

Figura III.6 - DFD explosão processo 3

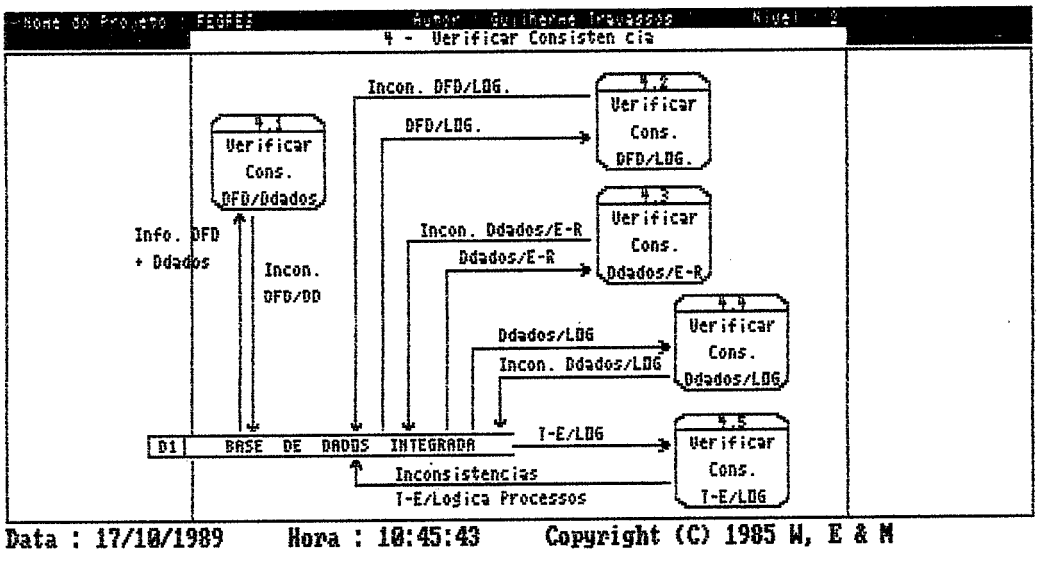

Figura III.7 - DFD explosão processo 4

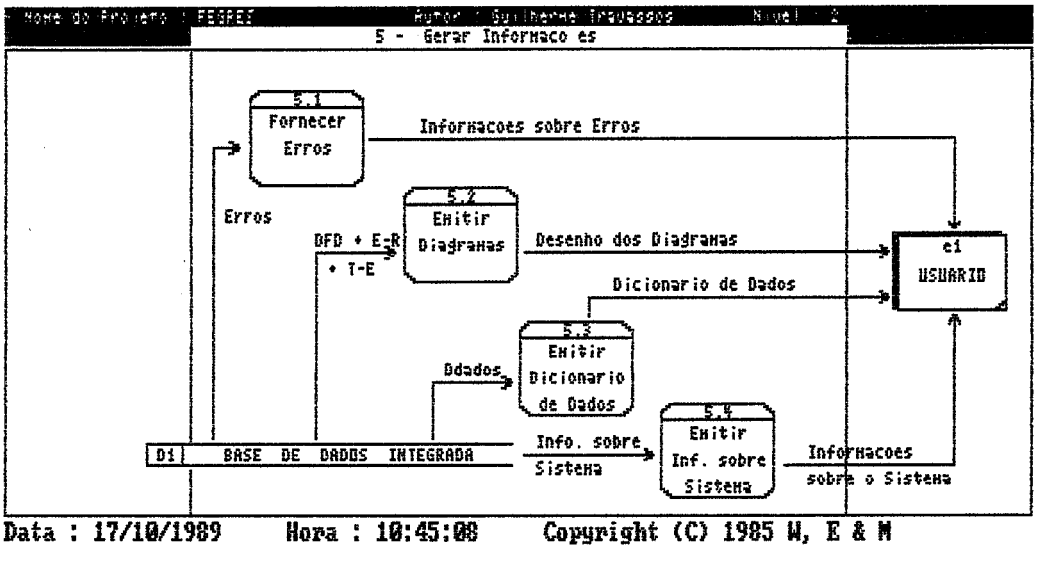

Figura III.8 - DFD explosão processo 5

Estes padrões de comunicação coletivamente definem a Interface com o Usuário (IU) que se mantém entre o usuário e a tecnologia do sistema de computação utilizado, NEWMAN [42]. Segundo BENNET [44] a qualidade da Interface com o Usuário depende do que o usuário vê (ou sente), o que o usuário deve saber para poder compreender o que é sentido, e que sequências de ações o usuário pode (ou deve) realizar para obter os resultados desejados. Na realidade, o principal requisito no projeto de uma IU é que ela torne o sistema fácil de ser usado pelo usuário e seja eficiente. Esta facilidade de uso pode ser obtida conjugando-se técnicas de interação apropriadas com os processos de cognição e coordenação motora. NEWMAN [42], FOLEY [45] , GARDINER [46], SCHNEIDERMAN [47]

HECKEL [48] e BERTINO [49] enumeram os seguintes fatores principais que caracterizam e qualificam uma interface de boa qualidade:

1) Tempo de Treinamento do Usuário: é necessário que a interface possua termos e conceitos específicos da área de aplicação do usuário, já que este não espera que a interface lhe obrigue a compreender termos e conceitos específicos da área de computação, e sim, que se comunique de uma maneira familiar com ele. A diminuição do tempo de aprendizado está diretamente ligada a este fator;

2) Tempo decorrido até que o usuário possa utilizar o sistema sem necessidade de ajuda: quando tarefas envolvem pensamento criativo, é aconselhável que o usuário seja capaz de realizar ações mecanicamente sem precisar pensar sobre esta ação. É preciso que o ato seja instintivo, pois o usuário está concentrado na tarefa que está tentando fazer, e não no mecanismo para executá-la;

3) Tempo para se Recuperar de Erros próprios ou do Equipamento: independente da aplicação é muito difícil que um usuário não cometa algum tipo de erro, principalmente se o mesmo não utiliza frequentemente a aplicação. A adoção de esquemas de ajuda, CLARK [50], que permitam uma rápida e segura recuperação de um erro é extremamente importante, e essencial para os

usuários ocasionais, pois a expectativa do usuário é que a aplicação não julgue seu uso, mas sim, que o ajude a resolver seus problemas de maneira segura e correta;

4) Motivação: a criação de uma barreira física ou mental por parte da aplicação é o primeiro passo para que o usuário fique insatisfeito em utilizá-la. O usuário espera sentir-se no controle do computador e da aplicação. Se conseguirmos atingir este objetivo, com certeza faremos com que o usuário fique motivado em usar cada vez mais a aplicação e a procurar novos caminhos para solucionar seus problemas a partir do computador;

5) Consistência: é extremamente importante que as aplicações utilizadas pelo usuário sejam consistentes, ou seja, possuam a mesma forma ou aparência nas estruturas de interações. É muito desagradável para o usuário ter que aprender a se comunicar com cada aplicação devido a não existência de um padrão pré-definido de comunicação. A falta de consistência provoca desmotivação, aumenta o tempo de aprendizagem e causa uma insatisfação generalizada por parte do usuário.

A obtenção das características anteriormente descritas para construção de FEGRES não é tarefa fácil. Além de envolver problemas específicos da área de computação, existe um reconhecimento de que pesquisas de fatores humanos e psicológicos podem ser úteis se elas forem dirigidas através da descoberta de modelos de pensamento humano os quais se aplicam a uma dada aplicação.

O padrão de interface com o usuário de FEGRES deve satisfazer os requisitos acima. Para que isto seja possível é necessário que se utilize um Sistema Gerenciador de Interface com o Usuário, SGIU, que garanta a consistência da interface entre as aplicações componentes do ambiente, facilidade de uso por parte do usuário e que forneça um ambiene adequado de desenvolvimento para a inclusão de novas aplicações. Um SGIU direciona e gerencia (controla) a interação do usuário num domínio de aplicação permitindo um desenvolvimento rápido e consistente. Ele pode ser visualizado também como uma ferramenta para aumentar a

produtividade e reduzir o custo de desenvolvimento de uma aplicação BETTS [51]. Esta melhoria no desenvolvimento pode ser obtida se considerarmos um SGIU como uma linguagem de 4a geração, onde a concentração de esforços é muito maior na especificação do produto do que na implementação.

Dois pontos de vista precisam ser considerados quando falamos de SGIU. O do desenvolvedor da aplicação e o do usuário. O desenvolvedor da aplicação enxerga um SGIU como uma ferramenta que lhe dará suporte para a construção de seu produto (interface) ao passo que o usuário espera que o SGIU proveja o suporte para uma fácil e efetiva utilização da aplicação.

Os fatores que influenciam diretamente um SGIU são:

1) Consistência: podemos dizer que consistência é sempre definida com respeito a alguma dimensão. É sempre julgada com respeito a qualidade (propriedades) de algum objeto quando comparado a qualidade de algum outro objeto adotado como padrão, BENETT [44]. O usuário espera que um SGIU mantenha consistência entre aplicações, ou seja, que determinada tarefa de interação possa sempre ser realizada através de técnica possuindo a mesma característica entre as aplicações desenvolvidas e que foi fornecida pelo SGIU ao desenvolvedor da aplicação;

2) Suporte para Usuários Novatos e Experientes: no projeto de uma interface temos que levar em consideração o nível de conhecimento do usuário, para não desanimarmos os usuários mais experientes com aplicações que subestimem sua capacidade intelectual e não desmotivar os usuários mais inexperientes com aplicações que não condizem com sua capacidade normal;

3) Suporte para Recuperação e Tratamento de Erros: um SGIU necessita fornecer estruturas robustas de recuperação e tratamento de erros para que torne o mais transparente possível para o usuário o processo de recuperação de erros cometidos durante a utilização da aplicação;

4) Suporte para Direcionalidade e Extensibilidade da Aplicação: aplicações não são estáticas e

se modificam com o tempo. O fornecimento pelo SGIU de facilidades que permitam estender, ou modificar, a aplicação de maneira simples e rápida faz com que reduzamos o custo de manutenção e consequentemente o custo de desenvolvimento, aumentando a produtividade de desenvolvimento,

5) Suporte para Ajuda e Assistência em Múltiplos Níveis: é extremamente benéfico para o usuário utilizar a aplicação para tirar dúvidas sobre a própria aplicação. Isto evita que a qualquer problema o usuário se veja obrigado a recorrer a manuais do sistema e outros dispositivos utilizados para documentar interações e atividades da aplicação;

6) Suporte de Direcionamento da Interface para o Usuário Final: a possibiliade de configuração dinâmica da interface permite personalizar as aplicações, adaptando características básicas para cada tipo de usuário.

III.2.3 Requisitos de Qualidade

A qualidade de um sistema está diretamente relacionada com a qualidade de sua especificação. Uma especificação de boa qualidade é obtida a partir do momento que o desenvolvedor da aplicação se utiliza corretamente de métodos apropriados para tal fim.

O primeiro passo para obtermos uma especificação de boa qualidade é construindo uma especificação consistente. Esta consistência deve existir entre todos os documentos gerados e é difícil de ser obtida se todos os documentos forem construídos de forma manual. Os documentos devem ser consistentes com a linguagem que foi utilizada para construí-los, devem se complementar, mostrando a mesma informação de variados pontos de vista, e não devem apresentar falta ou excesso de informações.

Não é possível para o desenvolvedor da aplicação construir uma especificação de boa qualidade sem um suporte adequado que permita gerenciar o processo de desenvolvimento sem interferir no processo como um todo, e que integre as informações de forma a garantir para o desenvolvedor que nada será esquecido de um documento para outro. Neste Sentido FEGRES deve suprir ao projetista facilidades apropriadas que permitam a construção de uma especificação com um nível de qualidade aceitável.

III.2.4 Requisitos de Desempenho

FEGRES é, na sua essência, um ambiente interativo. Toda comunicação do usuário com o ambiente é feita diretamente através da manipulação de objetos contidos na sua interface e que fazem um mapeamento direto com as entidades que os mesmos representam. É necessário, portanto, que o usuário não se sinta frustado ao utilizar as ferramentas. A resposta as consultas realizadas pelo usuário devem ser rápidas, como se o mesmo estivesse utilizando um editor de textos convencional, e devem ser devidamente comunicadas sobre seu andamento para não causar uma expectativa ainda maior por parte do usuário. Por outro lado, transações realizadas com FEGRES podem ser longas. Tratar um objeto envolve a manipulação de uma série de características complementares que provocam um certo retardo no tempo de processamento. Além do mais, todo um processo de gerenciamento de informação deve ser realizado para que o conjunto de informações que formam a especificação seja mantido o mais consistente possível.

Neste sentido três aspectos devem ser considerados para que FEGRES possibilite um alto poder de processamento num tempo mínimo. O que diz respeito ao gerenciamento da interface com o usuário, o que diz respeito com o gerenciador de base de dados,e o que diz respeito com a aplicação propriamente dita. Precisamos portanto ter um bom gerenciador de interface com o usuário, um gerenciador de base de dados de alto desempenho e ao mesmo tempo dotado de características que possibilitem o tratamento de objetos pertencentes a ferramenta, e que a

aplicação seja construída no mais alto padrão de Engenharia de Software visando obter o melhor desempenho no melhor estilo de construção.

III.2.5 Requisitos de Armazenamento de Informações

Para que FEGRES consiga fornecer ao desenvolvedor da aplicação uma especificação consistente é preciso que todas as ferramentas componentes do ambiente compartilhem das mesmas informações. Um SGBD, Sistema Gerenciador de Base de Dados, pode fornecer um gerenciamento apropriado das informações e ao mesmo tempo facilitar a inclusão de novas ferramentas no ambiente. A maioria dos SGBD's baseiam-se nos modelos de rede, hierárquico, relacional e semântico, DATE [52], sendo que os mais potentes são baseados, direta ou indiretamente, no modelo relacional. Entretanto, apesar de toda sua versatilidade e eficiência, são inadequados para suportar um ambiente de desenvolvimento de software, BERSTEIN [53].

O motivo desta inadequação é que a maioria dos sistemas é construída para ser utilizada em áreas adminstrativas, onde a estrutura do modelo é estática e não envolve trocas rápidas nos dados, não provendo facilidades para tratamento de aplicações complexas, onde a representação da informação difere radicalmente daquela para que foram projetados, PENEDO [2], FOISSEAU[54], NASKIN[55]. Extensões aos modelos convencionais precisam ser acrescidas para que um sistema deste tipo possa suportar um ambiente de desenvolvimento de software.

Os objetos manipulados por FEGRES são , em sua maioria, construídos de forma a proporcionar uma representação a mais próxima possível do mundo real. Esta forma de construção traz a necessidade de possuirmos um sistema gerenciador de base de dados específico, que de alguma forma, sane as principais deficiências encontradas em gerenciadores convencionais. Além do mais, FEGRES é uma ferramenta com expectativas de crescimento, podendo incorporar a seu contexto novas aplicações que aumentem sua abrangência na

construção de um produto de software.

As seguintes facilidades devem ser observadas na escolha de um sistema gerenciador de base de dados para FEGRES:

1) Ambiente Computacional: a arquitetura dos ambientes computacionais está se modificando. As aplicações antes concentradas em um único equipamento hoje tendem a ser processadas em redes de potentes estações de trabalho, conectadas entre si e servidas por máquinas cada vez mais poderosas. Esta arquitetura cria um ambiente de trabalho em que cada equipamento pode estar utilizando uma ferramenta diferente e acessando a mesma base de informações. É preciso que o gerenciador possua recursos para processamento distribuído visando facilitar ao ambiente que vários integrantes da equipe de desenvolvimento possam estar manuseando pedaços distintos da informação sem o risco de perda ou inconsistência;

2) Dados Complexos: dados, num contexto de Ambientes de Desenvolvimento de Software, são extremamente complexos. É preciso que exista mais flexibilidade no tipo de dado que pode ser armazenado e controlado por um SGBD. Recursos para criação dinamica de dados, inserção de novos tipos, tais como uniões, tipificação, arranjos e estruturas aninhadas é essencial para a representação de dados complexos;

3) Versões: durante o desenvolvimento de um projeto várias versões dos documentos são criadas e manuseadas pelos desenvolvedores da aplicação. Controlar estes documentos, proporcionando que documentos sejam agrupados com versões correspondentes é uma característica marcante e necessária para que um SGBD forneça um bom suporte para um ambiente de Desenvolvimento;

4) Transações: num Ambiente de Desenvolvimento de Software, transações são de longa duração e envolvem o bloqueio de um objeto por um período de tempo considerável, HITCHCOCK [58], LORIE [57]. Os controles de concorrência devem ser mais elaborados e separados do tratamento

normal de execução, e a criação de uma abordagem que seja mais compatível com os tipos de atividades que estão envolvidas no gerenciamento e desenvolvimento de projetos de software deve ser feita;

5) Visões: permitir que uma determinada aplicação acesse toda a base de dados provocará uma complicação a mais na execução de uma operação, pois será mais difícil encontrar o dado que a mesma desejar, além de ter implicações importantes de segurança e consistência da base de dados. Cada ferramenta existente no ambiente na verdade é uma visão diferente da base de dados, que mapeia os objetos de forma distinta conforme o objetivo de cada uma. Dessa forma é preciso que o SGBD possa controlar estas ferramentas fornecendo as informações necessárias de maneira rápida e segura, não importando quão complexas estas informações, ou visões, sejam.

III.2.6 Prioridades de Implementação

FEGRES é um conjunto de ferramentas abrangente que permitirá ao engenheiro de software visualizar o processo de desenvolvimento como um todo através de métodos estruturados de desenvolvimento. Estes métodos, por sua vez, são compostos por vários instrumentos e técnicas que possibilitam a implementação de ferramentas automatizadas para suportá-los. O objetivo de FEGRES nesta primeira versão é fornecer suporte ao engenheiro de software para a construção da fase de especificação de requisitos através da Análise Estruturada de Sistemas, GANE[14], juntamente com algumas extensões a este método.

Tendo em vista estas carcaterísticas, e também a necessidade de suprir FEGRES de um ambiente computacional adequado devem, nesta primeira etapa, ser implementados:

1) O Editor de Diagrama de Fluxo de Dados visando permitir uma representação funcional da aplicação que se está projetando;

2) O Editor de Diagramas de Entidade Relacionamento visando permitir uma representação relacional entre objetos da aplicação que se está projetando;

3) O Dicionário de Dados Estendido visando permitir o compartilhamento das informações entre os editores de diagramas e também ao armazenamento das fichas de documentação da aplicação.

CAPÍTULO IV

O PROJETO DE FEGRES

IV.1 Introdução

Conforme já apresentado no capítulo anterior, FEGRES em sua primeira versão preocupa-se com a fase de especificação do ciclo de desenvolvimento e para isto se utiliza de um conjunto de ferramentas que apoiarão o engenheiro de software na construção dos documentos que comporão esta especificação.

Foram escolhidas para implementação os editores de diagramas de Fluxo de Dados e Entidades-Relacionamentos juntamente com o Dicionário de Dados devido a:

1) utilização de forma manual das técnicas acima são de difícil gerenciamento e manutenção;

2) representarem o início de todo o processo de desenvolvimento, pois através das informações geradas por elas é que se poderá prosseguir com o projeto;

3) possibilidade de integração de novas ferramentas aproveitando o ambiente que passará a existir;

4) obter uma validação e aceitação por parte do usuário da interface projetada, devido as três ferramentas possuirem características distintas e ao mesmo tempo que melhor representam futuras expansões.

O editor de Diagramas de Fluxo de Dados corresponde à implementação do processo "Desenhar DFD" e "Emitir Diagrama" apresentados nas figuras III.4 e III.8. Tem como objetivos auxiliar o usuário na construção, alteração, eliminação e impressão dos Diagramas de Fluxo de Dados.

O editor de Diagramas de Entidades-Relacionamentos corresponde à implementação do processo "Desenhar DER" e "Emitir Diegrama" apresentados nas figuras III.4 e III.8. . Tem como objetivos auxiliar o usuário na construção, alteração, eliminação e impressão

dos Diagramas de Entidades-Relacionamentos.

O Dicionário de Dados corresponde à implementação dos processos " Completar DDADOS" e "Emitir Dicionário de Dados" apresentados nas figuras III.5 e III.8.. Tem como objetivos auxiliar o usuário na inclusão, alteração, eliminação, edição e impressão das fichas que descrevem os objetos pertencentes aos diagramas de Fluxo de Dados e Entidades-Relacionamentos.

Neste capítulo são descritos:

.recursos necessários;

.projeto físico dos objetos;

.algoritimos de acesso aos objetos.

IV.2 Recursos Necessários

IV.2.1 Hardware

São considerados requisitos mínimos de hardware para que FEGRES possa ser utilizado sem detrimento de suas caracterísitcas básicas:

- microcomputador compatível com o IBM-PC AT;
- mínimo de 640 kbytes de memória RAM;
- 1 Disco rígido do tipo winchester de no mínimo 20 Mbytes;
- 1 impressora gráfica;
- 1 placa gráfica padrão EGA;
- 1 monitor de vídeo para placa gráfica padrão EGA;
- 1 mouse .

IV.2.2 Software

- Sistema Gerenciador de Interface

Devido as características necessárias em FEGRES para proporcionar ao usuário uma interface que seja fácil de ser utilizada e aprendida e, ao mesmo tempo , robusta e flexível escolheu-se o sistema WINDOWS 2.0, desenvolvido pela MICROSOFT [25], como o padrão de

interface com o usuário para as ferramentas componentes de FEGRES. O uso do Windows como um padrão de interface e ambiente de desenvolvimento possibilita uma série de vantagens. Tanto o usuário como o desenvolvedor da aplicação tiram algum tipo de proveito através da utilização deste ambiente. Para a escolha do WINDOWS 2.0 levou-se em consideração os seguintes critérios:

1) Interface com o Usuário consistente entre aplicações: independente da aplicação desenvolvida para o WINDOWS, a aparência da interface é sempre a mesma, facilitando o aprendizado por parte do usuário. Esta consistência é obtida devido ao fato das aplicações especialmente construídas para ele se utilizarem sempre das rotinas existentes nos módulos de controle do Windows para realizar suas operações;

2) Programas funcionam naturalmente: o usuário gerencia símbolos ao invés de palavras. A aplicação funciona num ambiente orientado ao objeto, tornando seu uso mais natural. A disponibilidade de diversos recursos torna o ambiente rico em metáforas apropriadas para causar a predisposição psicológica do usuário para a situação que determinado objeto deseja provocar;

3) Independência de dispositivo: a máquina virtual do WINDOWS permite que as aplicações construídas para ela sejam independenetes de dispositivo, facilitando a manutenção e incremento do ambiente. A incorporação de novas características nos dispositivos periféricos interligados ao ambiente implica apenas na substituição, por parte do fabricante do dispositivo, do driver responsável pelo seu gerenciamento, não necessitando que as aplicações sejam modificadas;

4) Suporte a multi-tarefa: aplicações desenvolvidas para o WINDOWS conseguem compartilhar recursos satisfatoriamente, através da utilização dos métodos de gerenciamento fornecidos pelo ambiente. O perfeito compartilhamento dos recursos só é possível a partir de aplicações que sejam especialmente projetadas para o Windows e levando em consideração estes recursos;

5) Gerenciamento de memória: não importa a quantidade de memória necessária para uma aplicação, o WINDOWS gerencia este recurso de forma transparente para o usuário. Com isto as aplicações não têm, a princípio, limites de tamanho de memória para executar;

6) Possibilidade de comunicação entre processos: é possível construir aplicações que se comunicam entre si, trocando informações dinamicamente durante a execução, através de troca de mensagens, bibliotecas dinâmicas, área de dados comum e protocolos de comunicação apropriados;

7) Uso de técnicas modernas de programação: WINDOWS já incorpora em seu ambiente características de programação orientada ao objeto, possibilitando a utilização por parte do desenvolvedor de técnicas modernas de programação;

8) Alta portatilidade: WINDOWS pode ser executado em qualquer equipamento que consigam operar sob sistema operacional DOS facilitando a portatilidade do ambiente para diversas máquinas;

10) Atualidade: WINDOWS pode ser considerado o estado da arte para ambientes de interface baseados em equipamentos IBM-PC compatíveis e PS/2.

Infelizmente a utilização do WINDOWS traz algumas desvantagens que devem ser destacadas:

1) Tempo de execução: devido a todo o processo gerencial existente por parte do WINDOWS para controlar seu ambiente, requerendo em alguns casos equipamentos computacionais mais poderosos, o desempenho da aplicação é um pouco menor do que uma aplicação convencional;

2) Complexidade de desenvolvimento: a utilização de conceitos modernos e técnicas de programação apropriadas e a existência de uma extensa biblioteca de funções e tipos de mensagens aumentam o tempo de desenvolvimento, diminuindo a produtividade dos integrantes da equipe. Para as primeiras aplicações o tempo de aprendizado para o desenvolvedor também é um pouco maior do que para aplicações convencionais e exige projetistas e programadores experientes. POTTER [58], PETZOLD [59]

3) Suporte para aplicações de tempo real: programas executando sob o WINDOWS estão sob seu controle, não podendo sofrer interrupções externas. Deste modo o ambiente não é apropriado para o desenvolvimento de aplicações de tempo real.

- Linguagem de desenvolvimento

Apesar de existirem uma série de linguagens disponíveis atualmente que poderiam ser utilizadas na construção de FEGRES o uso do WINDOWS como sistema de interface restringe o universo das linguagens disponíveis para PASCAL, ASSEMBLER e C.

A escolha de C , KERNIGHAN [60], MICROSOFT [61], como linguagem de programação levou em consideração as seguintes caracterísitcas:

1) ser uma linguagem extremamente poderosa e robusta apresentando um desempenho satisfatório e produzindo um código otimizado e seguro;

2) ser extremamente portável entre os vários equipamentos IBM-PC compatíveis disponíveis no mercado;

3) possuir recursos de fácil manipulação de tipos de dados complexos, tais como uniões, arranjos, listas, dentre outros, tornando simples a programação das funções de criar, alterar e eliminar estes tipos através de alocação dinâmica de memória;

4) ser própria para programação de sistemas estruturados;

5) possuir comandos simples e poderosos que facilitam a manutenção do sistema;

6) possibilitar programação em baixo nível no contexto de uma linguagem de alto nível;

7) ser a que melhor se relaciona com o conjunto de funções fornecidos pelo ambiente WINDOWS.

- Sistema Gerenciador de Base de Dados

Conforme descrito no capítulo anterior é imprescindível que FEGRES possua um gerenciador de base de dados que possa suportar suas necessidades de armazenamento e tratamento das informações. Após várias análises de sistemas gerenciadores disponíveis não foram encontrados no mercado SGBD's que pudessem ser utilizados como gerenciadores de

objetos complexos sob ambiente operacional WINDOWS. Neste sentido optou-se pela construção de um pequeno gerenciador de base de dados baseado no modelo relacional e inspirado na arquitetura do COPPEREL, SOUZA [62].

A construção de um pequeno sistema gerenciador de base de dados específico para armazenamento e tratamento dos objetos de FEGRES traz as seguintes vantagens:

1) ser um sistema projetado especificamente para o ambiente, melhorando assim o desempenho das aplicações;

2) poder utilizar recursos disponíveis no WINDOWS na sua construção;

3) possibilitar o total domínio da aplicação, favorecendo com isso que novas características sejam incorporadas ao sistema;

4) fornecer subsídios para a adaptação do COPPEREL às novas necessidades de tratamento e recuperação de informação , possibilitando que em um futuro próximo se possa contar com um sistema robusto como o COPPEREL rodando sob ambiente WINDOWS.

IV.3 Projeto Físico dos Objetos

IV.3.1 Objetos básicos Tratados por FEGRES

Na figura IV.1 podemos visualizar o modelo de dados de FEGRES. Os objetos representados são os que aparecem na definição das ferramentas anteriormente citadas e que precisam ser implementadas. A manutenção das informações relativas a cada objeto e que são criadas com a utilização das ferramentas anteriormente citadas necessitam de um sistema gerenciador de base de dados robusto e confiável para seu efetivo controle. Apesar de possuirmos vários SGBD's no mercado, e além disto, dispormos de tecnologia própria, através do SGBD COPPEREL, ainda assim a utilização do WINDOWS como ambiente de interface cria uma série de restrições de execução que necessitam ser incorporadas aos sistemas existentes para que executem perfeitamente sob este novo ambiente. Devido a isto um pequeno núcleo de um gerenciador baseado na arquitetura do COPPEREL foi construído para realizar o controle básico dos objetos.

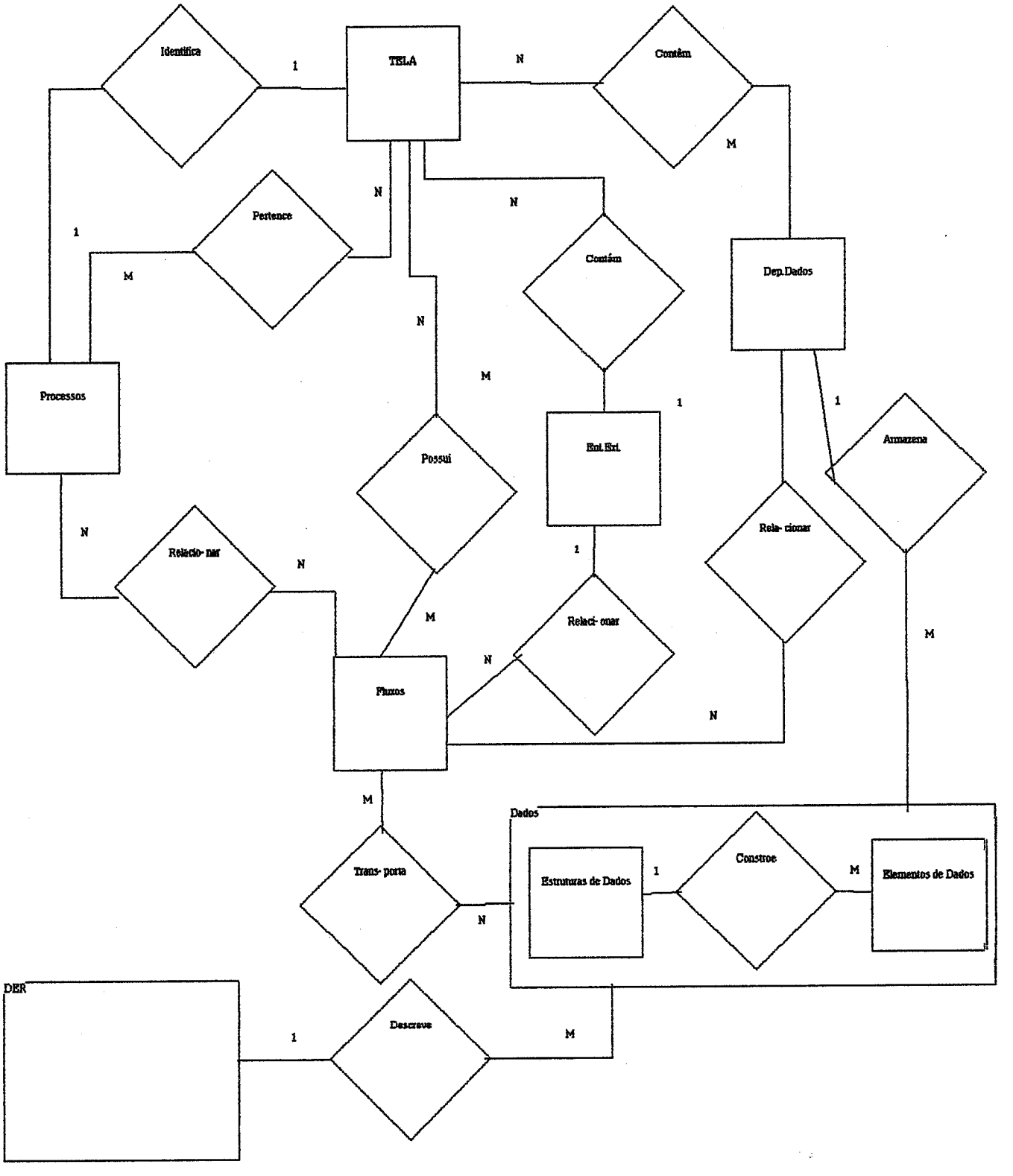

Figura IV.1 - O Modelo de Dados de FEGRES

IV.3.2 Restrições de Integridade dos Objetos

1) PROCESSOS

1.1) Todo PROCESSO possui um CRIADOR (Contexto, nível 1, etc);

1.2) Não pode existir mais de um PROCESSO com o mesmo CRIADOR e a mesma identificação;

1.3) Todo PROCESSO possui pelo menos um FLUXO entrando e um saindo;

1.4) Mover um PROCESSO implica em mover os FLUXOS associados;

1.5) Remover um PROCESSO implica em marcar os FLUXOS associados como inválidos,remover todos os seus filhos e remover sua ficha. Esta remoção só pode ocorrer no nível em que foi criado;

1.6) Um PROCESSO pode aparecer em três níveis distintos: Nível de criação, Explodido, externo a uma explosão;

1.7) Explodir um PROCESSO implica em apresentar todos os elementos relacionados externos a explosão.

2) ENTIDADE EXTERNA

2.1) Toda ENTEXTERNA possui uma identificação e só pode ser criada em nível de Contexto;

2.2) Uma ENTEXTERNA pode aparecer repetidamente num DFD em diversas telas;

2.3) Mover uma ENTEXTERNA implica em mover todos os FLUXOS associados aquela repetição da ENTEXTERNA;

2.4) Remover uma ENTEXTERNA implica em marcar os fluxos associados como inválidos relativos aquela repetição da ENTEXTERNA. Caso aparição seja a última, implica em remover sua ficha;

2.5) Uma ENTEXTERNA não pode ser representada dentro da área de explosão de um PROCESSO.

3) DEPÓSITOS DE DADOS

3.1) Todo DEPÓSITO possui um CRIADOR e uma identificação;

3.2) Um DEPÓSITO pode aparecer repetidamente num DFD em diversas telas;

3.3) As informações contidas num DEPÓSITO tem que ser aquelas que chegam através dos FLUXOS que entram;

3.4) Todo DEPÓSITO possui pelo menos um FLUXO entrando e um saindo;

3.5) Mover um DEPÓSITO implica em mover todos os FLUXOS associados com aquela repetição do DEPÓSITO;

3.6) Remover um DEPÓSITO implica em marcar todos os fluxos associados com aquela repetição como inválidos. Caso seja última repetição, retirar ficha do DEPÓSITO;

3.7) Um DEPÓSITO externo a um PROCESSO não pode ser representado na área de explosão deste PROCESSO;

3.8) Um DEPÓSITO interno a um PROCESSO não pode ser representado fora da área de explosão deste PROCESSO.

4) FLUXOS DE DADOS

4.1) Todo FLUXO possui um CRIADOR e um nome;

4.2) Não podem ocorrer fluxos distintos com o mesmo nome;

4.3) Um FLUXO pode aparecer repetidas vezes no DFD com a mesma identificação mas com origem/destino diferentes;

4.4) Mover o elemento origem do FLUXO implica em mover todos os vértices do FLUXO menos o vértice destino;

4.5) Mover o elemento destino do FLUXO implica em mover todos os vértices do FLUXO menos o vértice origem;

4.6) Apenas as seguintes combinações de origem/destino são permitidas:

destino/origem	PROCESSO	ENT. EXTERNA	DEPÓSITO
PROCESSO	X	X	X
ENT. EXTERNA	X	-	-
DEPÓSITO	X	-	-

5) ENTIDADE

5.1) Uma ENTIDADE só pode aparecer uma vez no DER;

5.2) Uma ENTIDADE deve possuir um NOME;

5.3) Uma ENTIDADE deve possuir atributos;

5.4) Os ATRIBUTOS de uma ENTIDADE são elementos existentes no Dicionário de Dados;

5.5) Uma ENTIDADE é considerada FRACA quando sua existência depende da existência de outra ENTIDADE. É considerada FORTE caso contrário;

6) RELACIONAMENTOS

6.1) Um RELACIONAMENTO pode aparecer apenas uma vez no DER;

6.2) Deve possuir um NOME;

6.3) Um RELACIONAMENTO pode possuir ATRIBUTOS;

6.4) Caso um RELACIONAMENTO possua ATRIBUTOS estes devem estar representados no Dicionário de Dados;

7) AGREGAÇÃO

7.1) Uma AGREGAÇÃO pode aparecer apenas uma vez no DER;

7.2) Deve possuir um NOME;

7.3) AGREGA ENTIDADES e RELACIONAMENTOS, juntamente com as ligações entre eles.

8) LIGAÇÕES

8.1) Uma LIGAÇÃO deve possuir uma ORIGEM e um DESTINO;

8.2) Apenas as seguintes combinações de origem/destino são permitidas:

Origem/Destino	ENTIDADE	RELACIONAMENTO	AGREGAÇÃO
ENTIDADE	-	X	-
RELACIONAMENTO	X	-	X
AGREGAÇÃO	-	X	-

8.3) Uma LIGAÇÃO deve possuir uma CARDINALIDADE e um indicador de TOTALIDADE.

IV.3.3 Tabelas para Representação dos Objetos

A partir do modelo de dados apresentado na figura (IV.1) e considerando-se as ferramentas a serem implementadas determinou-se a representação deste modelo a partir de tabelas. Estas tabelas são controladas e gerenciadas pelo próprio ambiente através da utilização do sistema gerenciador construído para tal fim. Após análise do modelo de dados chegou-se às seguintes tabelas:

1) Tabela PROCESSOS: armazena a ficha que descreve um processo na especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM.
IDENT_P	identificação do processo na especificação	alfa	10
NOME	identificação funcional do processo na especificação	alfa	128
DESCRIÇÃO	descrição resumida do processo	alfa	254
CRIADOR	Processo pai deste processo	alfa	10
ÚLTIMA INSTÂNCIA	última ocorrência do processo na especificação	numérico	2
RESUMO LÓGICO	apresentação resumida da lógica do processo	alfa	254
REFS. FÍSICAS	referências físicas do processo	alfa	128
DET. LÓGICA	local onde podem ser encontrados maiores detalhes sobre o processo	alfa	128

Comprimento Total: 914 bytes

Chave Primária: IDENT_P

Forma: PROCESSOS(IDENT_P, NOME, DESCRIÇÃO, CRIADOR, ÚLTIMA INSTÂNCIA, RESUMO LÓGICO, REFS. FÍSICAS,DET. LÓGICA)

2) Tabela Depósitos: armazena a ficha que descreve um depósito de dados na especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
IDENT_D	identifica um depósito de dados na especificação	alfa	5
NOME	nome do depósito de dado	alfa	30
DESCRIÇÃO	descreve resumidamente o depósito de dados	alfa	128
CRIADOR	Processo pai do depósito de dados	alfa	10
ÚLTIMA INSTÂNCIA	última ocorrência do depósito de dados na especificação	numérico	2
ACESSO IMEDIATO	elementos armazenados que permitem o acesso imediato a este depósito	alfa	128
ORG. FÍSICA	forma de organização do depósito de dados	alfa	128

Comprimento Total: 431 bytes

Chave Primária: IDENT_D

Forma: DEPÓSITOS(IDENT_D, NOME, DESCRIÇÃO, CRIADOR, ÚLTIMA INSTÂNCIA, ACESSOIMEDIATO, ORG.FÍSICA)

3) Tabela Fluxos: armazena as fichas que descrevem os fluxos de dados pertencentes a especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_FLUXO	identifica o fluxo de dados na especificação	alfa	40
DESCRIÇÃO	descreve resumidamente o fluxo de dados	alfa	128
CRIADOR	Processo pai do fluxo de dados	alfa	10
ÚLTIMA INSTÂNCIA	última ocorrência do fluxo de dados na especificação	numérico	2
DESCRIÇÃO AMPLIADA	descreve em maior detalhe o fluxo de dados	alfa	254
VOLUME	representa o volume de informação que o fluxo de dados pode transportar	alfa	128
TIPO	indica o número de setas que a representação gráfica do fluxo deve possuir	numérico	2

Comprimento Total: 564 bytes

Chave Primária: NOME_FLUXO

Forma: FLUXOS(NOME_FLUXO, DESCRIÇÃO, CRIADOR, ÚLTIMA INSTÂNCIA, DESCRIÇÃO AMPLIADA, VOLUME, TIPO) TIPO {SIMPLES, DUPLO}

4) Tabela EntExternas: armazenam as fichas de descrição das Entidades Externas existentes na especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
IDENT_E	identifica uma Entidade Externa na especificação	alfa	5
NOME	nome da Entidade Externa	alfa	40
DESCRIÇÃO	descreve o que a Entidade Externa representa na especificação	alfa	254
CRIADOR	Processo pai da Entidade Externa	alfa	10
ÚLTIMA INSTÂNCIA	última ocorrência da Entidade Externa na especificação	numérico	2
LINGUAGEM	linguagem de implementação, caso Entidade Externa seja um sistema	alfa	40
HARDWARE	tipo de hardware que Entidade Externa executa, caso Entidade Externa seja um sistema	alfa	40
PESSOAPD	responsável de processamento de dados pela Entidade Externa, caso Entidade Externa seja um sistema	alfa	40
FONEPD	telefone para contato com o responsável pelo processamento de dados, caso Entidade Externa seja um sistema	alfa	11
SETOR	departamento que desempenha o papel da Entidade Externa	alfa	128
PESSOAORG	responsável pela Entidade Externa no Setor correspondente	alfa	40
CARGO	cargo do responsável pela Entidade Externa	alfa	40
FONEOR	telefone para contato com responsável pela Entidade Externa numa organização	alfa	11

Comprimento Total: 661

Chave Primária: IDENT_E

Forma: ENTEXTERNAS(IDENT_E, NOME, DESCRIÇÃO, CRIADOR, ÚLTIMA INSTÂNCIA, LINGUAGEM, HARDWARE, PESSOAPD, FONEPD ,SETOR, PESSOAORG, CARGO, FONEORG)

5) Tabela Estruturas: armazenam as fichas que descrevem as estruturas de dados existentes numa especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_EST	nome da estrutura de dados	alfa	40
DESCRIÇÃO	descreve o que representa a estrutura de dados	alfa	254
VOLUME	informa qual o número de estruturas de dados podem ser manipulados pelo sistema especificado	alfa	128

Comprimento Total: 422 bytes

Chave Primária: NOME_EST

Forma: ESTRUTURAS(NOME_EST, DESCRIÇÃO, VOLUME)

6) Tabela Elementos: armazenam as fichas que descrevem os elementos de dados existentes na especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_ELE	nome do elemento de dados	alfa	40
DESCRIÇÃO	descreve o que o elemento de dados representa na especificação	alfa	254
TIPOELEMENTO	informa qual o tipo do elemento. Pode ser A,N, ou AN	alfa	2
DOMÍNIO	informa qual o domínio de aplicação do elemento de dados	alfa	128
VERTAMBÉM	informa o local onde podem ser encontradas maiores referências sobre o elemento de dados	alfa	128

Comprimento Total: 552 bytes

Chave Primária: NOME_ELE

Forma: ELEMENTOS(NOME_ELE, DESCRIÇÃO, TIPOELEMENTO, DOMÍNIO, VERTAMBÉM)

7) Tabela Glossário: armazena as fichas que descrevem os ítens de glossário existentes na especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_ITE	nome do ítem de glossário	alfa	40
DESCRIÇÃO	descreve o que o ítem de glossários representa na especificação	alfa	254
TIPOÍTEM	informa qual o tipo do ítem de glossário. Pode ser A,N, ou AN	alfa	2
DOMÍNIO	informa qual o domínio de aplicação do ítem de glossário	alfa	128
VERTAMBÉM	informa o local onde podem ser encontradas maiores referências sobre o ítem de glossário	alfa	128

Comprimento Total: 552 bytes

Chave Primária: NOME_ITE

Forma: GLOSSÁRIO(NOME_ITE, DESCRIÇÃO, TIPOÍTEM, DOMÍNIO, VERTAMBÉM)

8) Tabela EstrEstr: representa o relacionamento entre as estruturas de dados pertencentes a uma especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_EST	nome da estrutura de dados	alfa	40
NOME_EST_COMPÕEM	nome da estrutrura de dados componente	alfa	40

Comprimento Total: 80 bytes

Chave Primária: NOME_EST + NOME_EST_COMPÕEM

Chave Secundária: NOME_EST

Forma: ESTRESTR(NOME_EST, NOME_EST_COMPÕEM)

9) Tabela EstrElem: representa o relacionamento entre as estruturas de dados e os elementos de dados que a compõem.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_EST	nome da estrutura de dados	alfa	40
NOME_ELE	nome do elemento de dados	alfa	40

Comprimento Total: 80 bytes

Chave Primária: NOME_EST + NOME_ELE

Chave Secundária: NOME_EST

Forma: ESTRELEM(NOME_EST, NOME_ELE)

10) Tabela EstrFluxo: representa o relacionamento entre os fluxos de dados e as estruturas de dados que os mesmos transportam.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_FLUXO	nome do fluxo de dados	alfa	40
NOME_EST	nome da estrutura de dados	alfa	40

Compriment Total: 80 bytes

Chave Primária: NOME_FLUXO + NOME_EST

Chave Secundária: NOME_FLUXO

Forma: ESTRFLUXO(NOME_FLUXO, NOME_EST)

11) Tabela EleFluxo: representa o relacionamento entre os fluxos de dados e os elementos de dados que os mesmos transportam.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_FLUXO	nome do fluxo de dados	alfa	40
NOME_ELE	nome do elemento de dados	alfa	40

Comprimento Total: 80 bytes

Chave Primária: NOME_FLUXO + NOME_ELE

Chave Secundária: NOME_FLUXO

Forma: ELEFLUXO(NOME_FLUXO, NOME_ELE)

12) Tabela DepEstru: representa o relacionamento entre as estruturas de dados existentes na especificação e os depósitos de dados que as contém.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_DEPÓSITO	nome do depósito de dados	alfa	40
NOME_EST	nome da estrutura de dados	alfa	40

Comprimento Total: 80 bytes

Chave Primária: NOME_DEPÓSITO + NOME_EST

Chave Secundária: NOME_DEPÓSITO

Forma: DEPESTRU(NOME_DEPÓSITO, NOME_EST)

13) Tabela DepEle: representa os relacionamentos entre os elementos de dados existentes na especificação e os depósitos de dados que os contém.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_DEPÓSITO	nome do depósito de dados	alfa	40
NOME_ELE	nome do elemento de dados	alfa	40

Comprimento Total: 80 bytes

Chave Primária: NOME_DEPÓSITO + NOME_ELE

Chave Secundária: NOME_DEPÓSITO

Forma: DEPELE(NOME_DEPÓSITO, NOME_ELE)

14) Tabela Pseudônimo: armazenam os sinônimos que um elemento de dados pode assumir na especificação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_ELE	nome do elemento de dados	alfa	40
NOME	pseudônimo associado	alfa	40

Comprimento Total: 80 bytes

Chave Primária: NOME_ELE + NOME

Chave Secundária: NOME_ELE

Forma: PSEUDÔNIMO(NOME_ELE, NOME)

15) Tabela Valores: armazena os possíveis valores, juntamente com seu significado, que um elemento de dados de domínio discreto pode assumir.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_ELE	nome do elemento de dados	alfa	40
SIGNIFICADO	o que representa o valor associado	alfa	40
VALOR	valor que o elemento pode assumir	alfa	40

Comprimento Total: 120 bytes

Chave Primária: NOME_ELE + SIGNIFICADO

Chave Secundária: NOME_ELE

Forma: VALORES(NOME_ELE, SIGNIFICADO, VALOR)

16) Tabela InstProc: armazena os dados relativos a posição do objeto Processo na janela de apresentação do editor de diagrama de fluxo de dados.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
IDENT_P	identificação do processo no DFD	alfa	10
TELA	tela em que o desenho deve aparecer	numérico	2
ID_INST	número da instância do objeto	numérico	2
X	coordenada X da posição do objeto	numérico	2
Y	coordenada Y da posição do objeto	numérico	2

Comprimento Total: 18 bytes

Chave Primária: IDENT_P + TELA

Chave Secundária: ID_INST

Forma: INSTPROC(IDENT_P, TELA, ID_INST, X, Y)

16) Tabela InstEntExt: armazena os dados relativos ao objeto Entidade Externa na janela de apresentação do editor de diagrama de fluxo de dados.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
IDENT_E	identificação da Entidade Externa no DFD	alfa	5
TELA	tela em que o desenho deve aparecer	numérico	2
ID_INST	número da instância do objeto	numérico	2
X	coordenada X da posição do objeto	numérico	2
Y	coordenada Y da posição do objeto	numérico	2

Comprimento Total: 13 bytes

Chave Primária: IDENT_E + TELA

Chave Secundária: ID_INST

Forma: INSTENTEXT(IDENT_E, TELA, ID_INST, X, Y)

17) Tabela InstDep: armazena os dados relativos a aparência do objeto Depósito de Dados na janela de apresentação do editor de diagrama de fluxo de dados.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
IDENT_D	identificação do depósito de dados no DFD	alfa	5
TELA	tela em que o desenho deve aparecer	numérico	2
ID_INST	número da instância do objeto	numérico	2
X	coordenada X da posição do objeto	numérico	2
Y	coordenada Y da posição do objeto	numérico	2

Comprimento Total: 13 bytes

Chave Primária: IDENT_D + TELA

Chave Secundária: ID_INST

Forma: INSTDEP(IDENT_D, TELA, ID_INST, X, Y)

18) Tabela InstFluxo: armazena os dados relativos a aparência do objeto Fluxo de Dados na janela de apresentação do editor de diagrama de fluxo de dados.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_F	nome do fluxo de dados	alfa	40
TELA	tela a que pertence o objeto	numérico	2
ID_INST	número da instância do objeto	numérico	2
Xo	coordenada X da origem do fluxo de dados	numérico	2
Yo	coordenada Y da origem do fluxo de dados	numérico	2
TIPO_ORIGEM	tipo do objeto que é origem do fluxo de dados	alfa	1
ID_INST_O	número da instância do objeto de origem	numérico	2
Xd	coordenada X do destino do fluxo de dados	numérico	2
Yd	coordenada Y do destino da fluxo de dados	numérico	2
TIPO_DESTINO	tipo do objeto destino do fluxo de dados	alfa	1
ID_INST_D	número da instância do objeto destino	numérico	2
Xn	coordenada X do nome do fluxo	numérico	2
Yn	coordenada Y do nome do fluxo	numérico	2
TIPO_TRACADO	indicador para o percurso que o fluxo deve assumir	numérico	2

Comprimento Total: 64 bytes

Chave Primária: NOME_FLUXO + TELA

Chave Secundária: ID_INST

Forma: INSTFLUXO(NOME_F, TELA, ID_INST, Xo, Yo, TIPO_ORIGEM, ID_INST_O, Xd, Yd, TIPO_DESTINO, ID_INST_D, Xn, Yn, TIPO_TRAÇADO)

o->origem, d->destino, n->nome

19) Tabela Entidade: armazena a ficha, juntamente com a posição de apresentação, do objeto Entidade na janela de apresentação do editor de diagramas de entidades-relacionamentos.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_ENT	nome da entidade na especificação	alfa	40
X	coordenada x da posição do objeto	numérico	2
Y	coordenada y da posição do objeto	numérico	2
TIPO	tipo da Entidade. Pode ser FORTE ou FRACA	alfa	1

Comprimento Total: 45 bytes

Chave Primária: NOME_ENT

Forma: ENTIDADE(NOME_ENT, X, Y, TIPO)

20) Tabela Relacionamento: armazena a ficha, juntamente com a posição de apresentação, do objeto Relacionamento na janela de apresentação do editor de diagrama de entidade-relacionamento.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_REL	nome do relacionamento	alfa	40
X	coordenada x da posição do objeto	numérico	2
Y	coordenada y da posição do objeto	numérico	2

Comprimento Total: 44 bytes

Chave Primária: NOME_REL

Forma: RELACIONAMENTO(NOME_REL, X, Y)

21) Tabela Ligação: armazena a ficha, juntamente com a representação, do objeto Ligação na janela de apresentação do editor de diagrama de entidade-realcionamento.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
ORIGEM	nome do objeto origem da ligação	alfa	40
DESTINO	nome do objeto destino da ligação	alfa	40
Xo	coordenada x da origem da ligação	numérico	2
Yo	coordenada y da origem da ligação	numérico	2
TIPO_ORIGEM	tipo do objeto origem da ligação	alfa	1
Xd	coordenada x do destino da ligação	numérico	2
Yd	coordenada y do destino da ligação	numérico	2
TIPO_DESTINO	tipo do objeto destino da ligação	alfa	1
TOTALIDADE	indica se a ligação é Total ou Parcial	alfa	1
CARDINALIDADE	indica o grau da ligação	alfa	2
Xc	coordenada x da representação da Cardinalidade	numérico	2
Yc	coordenada y da representação da Cardinalidade	numérico	2
TIPO_TRACADO	indica qual o tipo de traçado da ligação	numérico	2

Comprimento Total: 100 bytes

Chave Primária: ORIGEM + DESTINO

Forma: LIGAÇÃO(ORIGEM, DESTINO, Xo, Yo, TIPO_ORIGEM, Xd, Yd,
TIPO_DESTINO, TOTALIDADE, CARDINALIDADE, Xc, Yc,
TIPO_TRAÇADO) c->cardinalidade

22) Tabela Agregação: armazena as fichas, juntamente com a posição, do objeto agregação na janela de apresentação do editor de diagramas de entidades-relacionamentos.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_AGR	nome da agregação	alfa	40
Xmax	coordenada x do vértice inicial da agregação	numérico	2
Ymax	coordenada y do vértice inicial da agregação	numérico	2
Xmin	coordenada x do vértice final da agregação	numérico	2
Ymin	coordenada y do vértice final da agregação	numérico	2
TEMVÉRTICE	indica se a agregação possui mais algum vértice para traçado	alfa	1

Comprimento Total: 49 bytes

Chave Primária: NOME_AGR

Forma: AGREGAÇÃO(NOME_AGR, Xmax, Ymax, Xmin, Ymin, TEMVÉRTICE)

23) Tabela EntAtrib: representa os atributos pertencentes a uma Entidade.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_ENT	nome da Entidade	alfa	40
NOME	nome do atributo	alfa	40
TIPO	tipo que o atributo pode assumir. Pode ser Dominante ou Genérico	alfa	1

Comprimento Total: 81 bytes

Chave Primária: NOME_ENT + NOME

Chave Secundária: NOME_ENT

Forma: ENTATRIB(NOME_ENT, NOME, TIPO) TIPO { DOMINANTE, GENÉRICO}

24) Tabela RelAtrib: representa os atributos pertencentes a um Relacionamento.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_REL	nome do Relacionamento	alfa	40
NOME	nome do atributo	alfa	40
TIPO	tipo que o atributo pode assumir. Pode ser Dominante ou Genérico.	alfa	1

Comprimento Total: 81 bytes

Chave Primária: NOME_REL + NOME

Chave Secundária: NOME_REL

Forma: RELATRIB(NOME_REL, NOME, TIPO)

25) Tabela VérticesAgregação: representa os vértices complementares para traçado de um Agregação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_AGR	nome da Agregação	alfa	40
ORDEM	ordem de traçado do vértice	numérico	2
X	coordenada x do vértice	numérico	2
Y	coordenada y do vértice	numérico	2

Comprimento Total: 48 bytes

Chave Primária: NOME_AGR + ORDEM

Chave Secundária: NOME_AGR

Forma: VÉRTICESAGREGAÇÃO(NOME_AGR, ORDEM, X, Y)

26) Tabela CompõemAgrega: representa os objetos componentes de uma Agregação.

ATRIBUTOS	DESCRIÇÃO	TIPO	TAM
NOME_AGR	nome da Agregação	alfa	40
NOME	nome do objeto componente	alfa	40
TIPO	tipo do objeto componente	alfa	1

Comprimento Total: 81 bytes

Chave Primária: NOME_AGR + NOME

Chave Secundária: NOME_AGR

Forma: COMPÕEMAGREGA(NOME_AGR, NOME, TIPO)

TIPO {ENTIDADE, RELACIONAMENTO}

IV.4 Principais Algoritimos para Tratamento dos Objetos

A seguir são apresentados os principais algoritmos que descrevem o comportamento dos objetos manuseados por FEGRES. Para sua descrição adotou-se uma forma de representação a mais próxima possível da sintaxe utilizada pela linguagem C.

1) OBJETO PROCESSO:

Algoritmo: CRIAR PROCESSO

Entradas: Posição do PROCESSO, Tela
Saídas: PROCESSO Incluído,erro

Início

```
Obter identificação do PROCESSO
 Selecionar da tabela PROCESSOS tal que  IDENT_P = Identificação
Se encontrou algum PROCESSO
 {
   Mensagem "PROCESSO já existente"
   retorne erro de inclusão
 }
senão
 {
      Acrescentar ficha de PROCESSO na tabela de PROCESSOS
      Acrescentar na tabela de INSTPROC entrada com IDENT_P = Identificação,
      TELA = Tela, {X, Y} = Posição
      Selecionar da tabela INSTFLUXOS tal que TELA = Tela & VÁLIDO = NÃO
```

```
                Enquanto houver FLUXOS faça
                {
                   Se {Xo,Yo} pertence  a área do PROCESSO
                         {
                         Se TIPO_DESTINO = (PROCESSO | | ENTEXTERNA
                            || DEPÓSITO)
                               Válido <- SIM
                          senão
                               Válido <- NÃO
                          Atualizar  entrada na tabela INSTFLUXOS  com
                         TIPO_ORIGEM = PROCESSO,   NOME_O = Ident_p,
                         VÁLIDO = Válido
                         }
                   senão
                      Se {Xd, Yd} pertence a área do PROCESSO
                            {
                               Se TIPO_ORIGEM = (PROCESSO || ENTEXTERNA ||
                                                          DEPÓSITO)
                                  Válido <- SIM
                               senão
                                  Válido <- NÃO
                               Atualizar  entrada na tabela INSTFLUXOS com
                               TIPO_DESTINO = PROCESSO,   NOME_D =  Ident_p,
                               VÁLIDO = Válido
                            }
                }
             }

retorne  Processo  Incluído
fim
```

Algoritimo: EXPLODIR PROCESSO

Entradas: Ident_p, Tela

Saídas: Explosao construida

Início

```
Selecionar da tabela INSTFLUXOS tal que TIPO_ORIGEM = PROCESSO &
IDENT_O = ident_p
faça enquanto houver FLUXOS {
      caso  TIPO_DESTINO
        {
          PROCESSO:
            {
              Selecionar da tabela INSTPROC tal que IDENT_P =
              INSTFLUXO-> IDENT_D & TELA =    Tela
              Se não achou
                {
                  Selecionar da tabela PROCESSO tal que
                  IDENT_P = INSTFLUXOS-> IDENT_D
                  Atualizar entrada com ULTIMA INSTÂNCIA <-
                  ULTIMA INSTÂNCIA + 1
                  Acrescentar entrada na tabela INSTPROC com
                  IDENT_P = INSTFLUXOS->IDENT_D, ID_INST = ULTIMA
                  INSTÂNCIA, X, Y, TELA = ident_p
                }
            }

          DEPÓSITO:
            {
              Selecionar da tabela INSTDEP tal que IDENT_DD =
               INSTFLUXO-> IDENT_D & TELA =Tela
```

```
        Se não achou
           {
              Selecionar da tabela DEPÓSITO tal que
              IDENT_DD  =  INSTFLUXOS->IDENT_D
              Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
               INSTÂNCIA + 1
              Acrescentar entrada na tabela INSTDEP com IDENT_DD =
              INSTFLUXOS->IDENT_D, ID_INST = ULTIMA INSTÂNCIA,
               X, Y, TELA = ident_p
           }
        }

ENTEXTERNA:
   {
      Selecionar da tabela INSTENTEXT tal que IDENT_E =
      INSTFLUXO-> IDENT_D & TELA = Tela
      Se não achou
         {
            Selecionar da tabela ENTEXTERNA tal que
            IDENT_E  =  INSTFLUXOS->IDENT_D
            Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
             INSTÂNCIA+ 1
            Acrescentar entrada na tabela INSTENTEXT com
            IDENT_E = INSTFLUXOS->IDENT_D,  ID_INST = ULTIMA
            INSTÂNCIA, X, Y, TELA = ident_p
         }
      }
   }
Selecionar da tabela INSTFLUXOS tal que NOME_F = INSTFLUXOS->
NOME_F  & TELA = Ident_p
Se não achou
   {
      Selecionar da tabela FLUXOS tal que NOME_F =
      INSTFLUXOS->NOME_F
      Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
```

```
                INSTÂNCIA + 1
                Acrescentar entrada INSTFLUXOS com NOME_F =
                INSTFLUXOS-> NOME_F, ID_INST = ULTIMA INSTÂNCIA,
                TIPO_ORIGEM = NULO, TIPO_DESTINO = INSTFLUXOS->
                TIPO_DESTINO
            }
        }

Selecionar da tabela INSTFLUXOS tal que TIPO_DESTINO = PROCESSO &
IDENT_O = ident_p
faça enquanto houver FLUXOS {
    caso TIPO_ORIGEM
      {
        PROCESSO:
          {
            Selecionar da tabela INSTPROC tal que IDENT_P =
            INSTFLUXO->IDENT_O & TELA= Tela
            Se não achou
              {
                Selecionar da tabela PROCESSO tal que
                IDENT_P = INSTFLUXOS-> IDENT_O
                Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
                 INSTÂNCIA +1
                Acrescentar entrada na tabela INSTPROC com IDENT_P =
                INSTFLUXOS->IDENT_O, ID_INST = ULTIMA INSTÂNCIA,
                 X, Y, TELA = ident_p
              }
            }

        DEPÓSITO:
          {
            Selecionar da tabela INSTDEP tal que IDENT_DD =
            INSTFLUXO->IDENT_O & TELA =Tela
```

```
        Se não achou
          {
            Selecionar da tabela DEPÓSITO tal que
            IDENT_DD  =  INSTFLUXOS->IDENT_O
            Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
            INSTÂNCIA + 1
            Acrescentar entrada na tabela INSTDEP com
             IDENT_DD =INSTFLUXOS->IDENT_O, ID_INST = ULTIMA
             INSTÂNCIA, X, Y, TELA = ident_p
          }
        }

ENTEXTERNA:
  {
    Selecionar da tabela INSTENTEXT tal que IDENT_E =
    INSTFLUXO->IDENT_O &  TELA = Tela
    Se não achou
      {
        Selecionar da tabela ENTEXTERNA tal que
        IDENT_E  =  INSTFLUXOS->IDENT_O
        Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
         INSTÂNCIA + 1
        Acrescentar entrada na tabela INSTENTEXT com
        IDENT_E = INSTFLUXOS->IDENT_O, ID_INST = ULTIMA
        INSTÂNCIA, X, Y, TELA = ident_p
      }
    }
  }
Selecionar da tabela INSTFLUXOS tal que NOME_F = INSTFLUXOS->
NOME_F  & TELA = ident_p
Se não achou
  {
    Selecionar da tabela FLUXOS tal que NOME_F =
    INSTFLUXOS->NOME_F
    Atualizar entrada com ULTIMA INSTÂNCIA <- ULTIMA
```

```
                    INSTÂNCIA + 1
                    Acrescentar entrada INSTFLUXOS com NOME_F =
                    INSTFLUXOS->NOME_F, ID_INST = ULTIMA INSTÂNCIA,
                    TIPO_DESTINO = NULO,TIPO_ORIGEM =
                    INSTFLUXOS->TIPO_ORIGEM
                    }
                }

                PEGAR EXPLOSÃO de ident_p
fim


Algoritimo: PEGAR EXPLOSÃO

Entradas: Ident_p
Saida:    Explosão
Início

    Selecionar da tabela INSTPROC tal que TELA = Ident_p
    faça enquanto houver PROCESSO
        {
          Selecionar da tabela PROCESSO tal que IDENT_P = INSTPROC->IDENT_P
          Explosão[i] <- Processo
        }

    Selecionar da tabela INSTFLUXO tal que TELA = Ident_p
    faça enquanto houver FLUXO
        Explosão[i] <- FLUXO

    Selecionar da tabela INSTDEP tal que TELA = Ident_p
    faça enquanto houver DEPÓSITO
        {
          Selecionar da tabela DEPÓSITO tal que IDENT_DD =
          INSTDEP->IDENT_DD
          Explosão[i] <- DEPÓSITO
```

```
        }

    Selecionar da tabela INSTENTEXT tal que TELA = Ident_p
    faça enquanto houver ENTEXTERNA
        {
            Selecionar da tabela ENTEXTERNA tal que IDENT_E =
            INSTENTEXT-> IDENT_E
            Explosão[i] <-ENTEXTERNA
        }

    retorne Explosão
fim


Algoritimo: REMOVER PROCESSO

Entradas: Ident_p, id_inst
Saídas:   Processo excluído, filhos excluídos

Início
    Selecionar da tabela INSTFLUXOS tal que TELA = Ident_p
    faça enquanto houver FLUXOS
        {
            Selecionar da tabela FLUXOS tal que NOME_F = INSTFLUXOS->NOME_F
            Se CRIADOR = Ident_p && Número de Instâncias = 0
                Retirar ficha da tabela FLUXOS
            Retirar entrada da tabela INSTFLUXOS
        }

    Selecionar da tabela INSTDEP tal que TELA = Ident_p
    faça enquanto houver DEPÓSITOS
        {
            Selecionar da tabela DEPÓSITOS tal que IDENT_DD =
            INSTDEP->IDENT_DD
            Se CRIADOR = Ident_p
```

```
            Retirar ficha da tabela DEPÓSITOS
            Retirar entrada da tabela INSTDEP
        }
    Selecionar da tabela INSTEXT tal que TELA = Ident_p
    faça enquanto houver ENTEXTERNA
        {
            Selecionar da tabela ENTEXTERNAS tal que IDENT_E =
            INSTEXTENT-> IDENT_E
            Se CRIADOR = Ident_p
                Retirar ficha da tabela ENTEXTERNAS
            Retirar entrada da tabela INSTENTEXT
        }

    Selecionar da tabela INSTFLUXOS tal que TIPO_ORIGEM = PROCESSO &
    IDENT_O = Ident_p & ID_INST_O = Id_inst
    faça enquanto houver FLUXOS
        Atualizar entrada na tabela INSTFLUXOS com TIPO_ORIGEM = NULO

    Selecionar da tabela INSTFLUXOS tal que TIPO_DESTINO = PROCESSO &
    IDENT_D = Ident_p & ID_INST_D = Id_inst
    faça enquanto houver FLUXOS
        Atualizar entrada na tabela INSTFLUXOS com TIPO_DESTINO = NULO

    Selecionar da tabela INSTPROC tal que TELA = Ident_p
    faça enquanto houver PROCESSOS
        REMOVER PROCESSO com INSTPROC->IDENT_P,INSTPROC->ID_INST

    Selecionar da tabela PROCESSO tal que IDENT_P = Ident_p
    Se CRIADOR = Tela
        Retirar ficha da tabela de PROCESSOS
    Selecionar da tabela INSTPROC tal que IDENT_P = Ident_p & ID_INST = Id_inst &
    TELA = Tela
    Retirar entrada da tabela INSTPROC
    retorne Processo e filhos excluídos
fim
```

2) OBJETO ENTIDADE EXTERNA:

```
Algoritimo: CRIAR ENTIDADE EXTERNA

Entradas: Posição da ENTEXTERNA, Tela
Saídas:   ENTEXTERNA incluída/atualizada, FLUXOS atualizados, erro

Início
    Se Pai > Nível 1 & Posição fornecida pertence a área de Explosão de PROCESSO
        {
          Mensagem "Posição de ENTEXTERNA inválida"
          retorne erro de inclusão
        }
    senão
        {
          Obter identificação da ENTEXTERNA
          Selecionar  da tabela ENTEXTERNAS tal que IDENT_E = Identificação
          Se não encontrou ENTEXTERNA
             Acrescentar ficha na tabela ENTEXTERNAS com
             IDENT_E = Identificação
           Acrescentar na tabela INSTENTEXT entrada com IDENT_E = Identificação,
          TELA = Tela,  {X,Y} = Posição
         Selecionar da tabela INSTFLUXOS tal que TELA = Tela & VÁLIDO = NÃO
             Enquanto houver FLUXOS faça
             {
               Se {Xo,Yo} pertence  a área da ENTEXTERNA
                {
                      Se TIPO_DESTINO = (PROCESSO )
                            Válido <- SIM
                       senão
                            Válido <- NÃO
                      Atualizar  entrada na tabela INSTFLUXOS  com
                      TIPO_ORIGEM = ENTEXTERNA, NOME_O = Ident_e,
```

```
                    VÁLIDO = Válido
                }
            senão
                Se {Xd, Yd} pertence a área da ENTEXTERNA
                {
                    Se TIPO_ORIGEM = (PROCESSO )
                        Válido <- SIM
                     senão
                        Válido <- NÃO
                    Atualizar  entrada na tabela INSTFLUXOS com
                    TIPO_DESTINO = ENTEXTERNA, NOME_D =  Ident_e,
                    VÁLIDO = Válido
                }
            }
        }
retorne  ENTEXTERNA  incluída
fim


Algoritimo:  REMOVER  ENTEXTERNA

Entradas:  Ident_e,  Id_inst,  Tela
Saídas:    ENTEXTERNA excluída

Início
    Selecionar da tabela INSTFLUXOS tal que TIPO_ORIGEM = ENTEXTERNA &
     IDENT_O = Ident_e & ID_INST_O = Id_Inst & TELA = Tela
    faça enquanto houver FLUXO
        {
            Selecionar da tabela INSTFLUXOS tal que NOME_F = Nome_f &
            IDENT_O =Ident_e  & TELA >= Tela
            faça enquanto houver FLUXOS
                {
                    Selecionar da tabela INSTENTEXT tal que IDENT_E = Ident_e &
                    ID_INST = INSTFLUXOS->ID_INST_O &
```

```
                TELA = INSTFLUXOS->TELA
                Retirar entrada da tabela INSTENTEXT
                Atualizar tabela INSTFLUXO com TIPO_ORIGEM = NULO
            }

        Atualizar entrada na tabela INSTFLUXOS com TIPO_ORIGEM = NULO
    }

Selecionar da tabela INSTFLUXOS tal que TIPO_DESTINO = ENTEXTERNA &
 IDENT_D = Ident_e & ID_INST_D = Id_Inst & TELA = Tela
faça enquanto houver FLUXO
    {
        Selecionar da tabela INSTFLUXOS tal que NOME_F = Nome_f &
        IDENT_D =Ident_e  & TELA >= Tela
        faça enquanto houver FLUXOS
            {
                Selecionar da tabela INSTDEP tal que IDENT_E = Ident_e &
                ID_INST = INSTFLUXOS->ID_INST_D & TELA = Tela
                Retirar entrada da tabela INSTENTEXT
                Atualizar tabela INSTFLUXO com TIPO_DESTINO = NULO
            }

        Atualiza entrada na tabela INSTFLUXOS com TIPO_DESTINO = NULO
    }

Selecionar da tabela INSTENTEXT tal que IDENT_E = Ident_e & ID_INST = Id_Inst
Retirar entrada da tabela INSTENTEXT
Selecionar da tabela INSTDEP tal que IDENT_E = Ident_e
Se não houver ENTEXTERNA
    {
        Selecionar da tabela ENTEXTERNAS tal que IDENT_E = Ident_e
        Retirar ficha da tabela ENTEXTERNA
    }
retorne  ENTEXTERNA  excluída
fim
```

3) OBJETO DEPÓSITO DE DADOS

Algoritimo: CRIAR DEPÓSITO DE DADOS

Entradas: Posição do DEPÓSITO, Pai

Saídas: DEPÓSITO Incluído/atualizado, erro

```
Início
    Se Pai > Nível 1 & Posição fornecida pertence a área de Explosão de PROCESSO
        {
          Mensagem "Posição de DEPÓSITO inválida"
          retorne erro de inclusão
        }
    senão
        {
          Obter identificação do DEPÓSITO
          Selecionar  da tabela DEPÓSITOS tal que IDENT_D = Identificação

          Se não encontrou DEPÓSITO
             Acrescentar ficha na tabela DEPÓSITOS com IDENT_D = Identificação
          Acrescentar na tabela INSTDEP entrada com IDENT_D = Identificação,
          TELA = Tela, PAI = Pai, {X,Y} = Posição, ORDEM = ORDEM + 1
          Selecionar da tabela INSTFLUXOS tal que TELA = Tela & VÁLIDO = NÃO
           Enquanto houver FLUXOS faça
             {
               Se {Xo,Yo} pertence  a área do DEPÓSITO
                   {
                    Se TIPO_DESTINO = (PROCESSO )
                          Válido <- SIM
                     senão
                          Válido <- NÃO
                    Atualizar  entrada na tabela INSTFLUXOS  com
                   TIPO_ORIGEM = DEPÓSITO, NOME_O = Ident_d,
```

```
                    VÁLIDO = Válido
                    }
                senão
                    Se {Xd, Yd} pertence a área do DEPÓSITO
                      {
                        Se TIPO_ORIGEM = (PROCESSO )
                            Válido <- SIM
                        senão
                            Válido <- NÃO
                        Atualizar  entrada na tabela INSTFLUXOS com
                        TIPO_DESTINO = DEPÓSITO, NOME_D =  Ident_d,
                        VÁLIDO = Válido
                      }
                    }
            }
retorne DEPÓSITO incluído
fim


Algoritimo:  REMOVER  DEPÓSITO

Entradas: Ident_dd, Id_inst, Tela
Saídas:    Depósito excluído

Início

    Selecionar da tabela INSTFLUXOS tal que TIPO_ORIGEM = DEPÓSITO &
    IDENT_O= Ident_dd & ID_INST_O = Id_Inst & TELA = Tela
    faça enquanto houver FLUXO
        {
          Selecionar da tabela INSTFLUXOS tal que NOME_F = Nome_f &
          IDENT_O = Ident_dd & TELA >= Tela
          faça enquanto houver FLUXOS
            {
              Selecionar da tabela INSTDEP tal que IDENT_DD = Ident_dd &
```

```
                ID_INST = INSTFLUXOS->ID_INST_O & TELA = INSTFLUXO->TELA
                Retirar entrada da tabela INSTDEP
                Atualizar tabela INSTFLUXO com TIPO_ORIGEM = NULO
            }
        Atualizar entrada na tabela INSTFLUXOS com TIPO_ORIGEM = NULO
    }
Selecionar da tabela INSTFLUXOS tal que TIPO_DESTINO = DEPÓSITO &
IDENT_D = Ident_dd  & ID_INST_D = Id_Inst & TELA = Tela
faça enquanto houver FLUXO
    {
        Selecionar da tabela INSTFLUXOS tal que NOME_F = Nome_f &
        IDENT_D = Ident_dd & TELA>= Tela
        faça enquanto houver FLUXOS
            {
                Selecionar da tabela INSTDEP tal que IDENT_DD = Ident_dd &
                ID_INST = INSTFLUXOS->ID_INST_D & TELA =
                INSTFLUXOS->TELA
                Retirar entrada da tabela INSTDEP
                Atualizar tabela INSTFLUXO com TIPO_DESTINO = NULO
            }
        Atualizar entrada na tabela INSTFLUXOS com TIPO_DESTINO = NULO
    }
Selecionar da tabela INSTDEP tal que IDENT_DD = Ident_d & ID_INST = Id_Inst
Retirar entrada da tabela INSTDEP
Selecionar da tabela INSTDEP tal que IDENT_DD = Ident_d
Se não houver DEPÓSITO
    {
        Selecionar da tabela DEPÓSITOS tal que IDENT_DD = Ident_d
        Retirar ficha da tabela DEPÓSITO
    }
retorne  Depósito  excluído
fim
```

4) OBJETO FLUXO DE DADOS

Algoritimo: CRIAR FLUXO DE DADOS

Entradas: Origem, Destino, Tela, TipoTraçado

Saídas: FLUXO Incluído, erro

```
Início
    Obter identificação do FLUXO
    Marcou_Origem <- FALSO
    Marcou_Destino <- FALSO
    Selecionar da tabela FLUXOS tal que NOME_F = Identificação
    Se não encontrou FLUXO
        Acrescentar na tabela de FLUXOS ficha com NOME_F = Identificação
    Acrescentar na tabela de INSTFLUXOS com NOME_F = Identificação, TELA = Tela,
    {Xo,Yo} = Origem, {Xd, Yd} = Destino, TIPO_TRAÇADO = TipoTraçado
    Selecionar da tabela de INSTPROC tal que TELA = Tela
    faça enquanto houver PROCESSOS
        {
          Se Origem pertence a área do PROCESSO
          {
            Atualizar tabela INSTFLUXOS com TIPO_ORIGEM = PROCESSO,
            NOME_O = Ident_p
            Marcou_Origem <- VERDADE
          }
        senão
          Se Destino pertence a área do PROCESSO
            {
              Atualizar tabela INSTFLUXOS com TIPO_DESTINO = PROCESSO,
               NOME_D = Ident_p
              Marcou_Destino <- VERDADE
            }
    Se ~(Marcou_Origem & Marcou_Destino)
        {
          Selecionar da tabela INSTDEP tal que TELA = Tela
```

```
        faça enquanto houver DEPÓSITOS  &
                ~(Marcou_Origem  &  Marcou_Destino)
           Se Origem pertence a área do DEPÓSITO & ~Marcou_Origem
              {
                 Atualizar  tabela INSTFLUXOS  com TIPO_ORIGEM = DEPÓSITO,
                 NOME_O = Ident_d
                 Marcou_Origem  <-  VERDADE
              }
           senão
              Se Destino pertence a área do DEPÓSITO & ~Marcou_Destino
                 {
                    Atualizar  tabela INSTFLUXOS  com TIPO_DESTINO =
                    DEPÓSITO, NOME_D =     Ident_d
                    Marcou_Destino  <-  VERDADE
                 }
        }
Se  ~(Marcou_Origem  &  Marcou_Destino)
     {
        Selecionar da tabela INSTENTEXT tal que  TELA = Tela
        faça enquanto houver ENTEXTERNAS  &
           ~(Marcou_Origem  &  Marcou_Destino)
           Se Origem pertence a área do DEPÓSITO & ~Marcou_Origem
              {
                 Atualizar  tabela INSTFLUXOS  com TIPO_ORIGEM =
                 ENTEXTERNA, NOME_O = Ident_e
                 Marcou_Origem  <-  VERDADE
              }
           senão
              Se Destino pertence a área da ENTEXTERNA & ~Marcou_Destino
                 {
                    Atualizar  tabela INSTFLUXOS  com TIPO_DESTINO =
                    ENTEXTERNA,NOME_D =Ident_e
                    Marcou_Destino  <-  VERDADE
                 }
```

```
            }
    }

Se ~(Marcou_Origem & Marcou_Destino)
     Atualizar entrada da tabela INSTFLUXOS com VÁLIDO = NÃO
senão
    Se (TIPO_ORIGEM || TIPO_DESTINO) ^ PROCESSO
        Atualizar entrada da tabela INSTFLUXOS com VÁLIDO = SIM
    senão
        Atualizar entrada da tabela INSTFLUXOS com Válido = NÃO

retorne FLUXO incluído
fim
```

```
Algoritimo: MUDAR TRACADO FLUXO

Entradas: Nome_f, Id_inst, Tipo_Traçado
Saídas: Instância do fluxo atualizada com novo traçado

Início

    Selecionar da Tabela INSTFLUXOS tal que NOME_F = Nome_f &. ID_INST = Id_inst
    Atualizar entrada com TIPO_TRACADO = Tipo_Tracado
    retorne Fluxo atualizado
fim
```

Algoritimo: REDIRECIONAR FLUXO

Entradas: Nome_f, Id_inst, Origem/Destino, Id_objeto, tipo_objeto ,id_inst_objeto, x,y

Saídas: Fluxo atualizado com nova origem/destino

```
Início
    Selecionar da Tabela INSTFLUXOS tal que NOME_F = Nome_f & ID_INST = Id_Inst
    Se Origem/Destino = ORIGEM
        {
            Caso tipo_objeto
                PROCESSO:
                   Atualizar entrada com ID_ORIGEM = id_objeto, TIPO_ORIGEM =
                   tipo, ID_INST_O = Id_inst_objeto, {Xo,Yo} = x,y

                ENTEXTERNA:
                DEPÓSITO:
                   Se INSTFLUXOS->TIPO_DESTINO != PROCESSO
                      {
                         Mensagem "Não posso redirecionar FLUXO"
                         retorne erro de atualizacao
                      }
                   senão
                      Atualizar entrada com ID_ORIGEM = id_objeto,
                      TIPO_ORIGEM = tipo,ID_INST_O = Id_inst_objeto, {Xo,Yo} =
                       x,y
          }
        senão
          {
            Caso tipo_objeto
                PROCESSO:
                   Atualizar entrada com ID_DESTINO = id_objeto, TIPO_DESTINO
                      = tipo,ID_INST_D =  Id_inst_objeto, {Xd,Yd} = x,y
```

```
                    ENTEXTERNA:
                    DEPÓSITO:
                       Se  INSTFLUXOS->TIPO_ORIGEM  !=  PROCESSO
                          {
                             Mensagem "Não posso redirecionar FLUXO"
                             retorne erro de atualização
                          }
                       senão
                          Atualizar entrada com ID_DESTINO = id_objeto,
                        TIPO_DESTINO = tipo, ID_INST_D = Id_inst_objeto, {Xd,Yd} = x,y
              }
      retorne FLUXO    redirecionado
fim
```

5) OBJETO ENTIDADE:

```
Algoritimo:  CRIAR  ENTIDADE

Entradas:  Posição
Saídas:      ENTIDADE incluída, erro

Início
      Obter Identificação da ENTIDADE
      Selecionar da tabela ENTIDADE tal que NOME_ENT = Identificação
      Se encontrou ENTIDADE
            {
               Mensagem "ENTIDADE já existe"
               retorne erro de inclusão
            }
      senão
```

```
        {
          Acrescentar ficha na tabela de ENTIDADE com NOME_ENT = Identificação,
          {X,Y} = Posição
          Obter ATRIBUTOS da ENTIDADE
          faça enquanto houver ATRIBUTOS
            Se ATRIBUTO é DETERMINANTE
              Acrescentar entrada na tabela ENTATRIB com NOME_ENT =
              Identificação, NOME = ATRIBUTO, TIPO = DETERMINANTE
            senão
              Acrescentar entrada na tabela ENTATRIB com NOME_ENT =
              Identificação, NOME = ATRIBUTO, TIPO = DETERMINANTE
        }

retorne ENTIDADE incluída
fim


Algoritimo: REMOVER ENTIDADE

Entradas: Posição
Saídas:   ENTIDADE excluída, erro

Início
      Se existir ENTIDADE nesta Posição
       {
          Pegar ficha da ENTIDADE
          faça enquanto houver LIGAÇÃO com ORIGEM = ENTIDADE
          {
              Pegar ficha da LIGAÇÃO
              retirar ORIGEM
              Invalidar ficha
                Guardar ficha daLIGAÇÃO invalidada
          }
          faça enquanto houver LIGAÇÃO com DESTINO = ENTIDADE
```

```
                {
                    Pegar ficha da LIGAÇÃO
                    retirar DESTINO
                    Invalidar ficha
                      Guardar ficha da LIGAÇÃO invalidada
                }
              retirar  ficha da ENTIDADE
              retorne ok
          }
        senão
          {
               Mensagem "ENTIDADE não encontrada"
              retorne erro exclusão
          }
fim
```

6) OBJETO RELACIONAMENTO:

```
Algoritimo:  CRIAR  RELACIONAMENTO
Entradas:  Posição
Saídas:      RELACIONAMENTO incluído, erro

Início
      Obter  Identificação  do  RELACIONAMENTO
      Selecionar  da  tabela  RELACIONAMENTO  tal  que  NOME_ENT  =  Identificação
      Se  encontrou  RELACIONAMENTO
            {
               Mensagem  "RELACIONAMENTO  já  existe"
               retorne  erro  de  inclusão
            }
      senão
```

```
        {
          Acrescentar ficha na tabela de RELACIONAMENTO com NOME_REL=
          Identificação, {X,Y} = Posição
          Obter ATRIBUTOS do RELACIONAMENTO
          faça enquanto houver ATRIBUTOS
            Se ATRIBUTO é DETERMINANTE
              Acrescentar entrada na tabela RELATRIB com NOME_REL =
              Identificação, NOME = ATRIBUTO, TIPO = DETERMINANTE
            senão
              Acrescentar entrada na tabela RELATRIB com NOME_REL =
              Identificação, NOME = ATRIBUTO, TIPO = DETERMINANTE
        }

retorne  RELACIONAMENTO  incluído
fim


Algoritimo:  REMOVER  RELACIONAMENTO

Entradas:  Posição
Saídas:    RELACIONAMENTO excluído erro

Início
      Se existir RELACIONAMENTO nesta Posição
      {
        Pegar ficha do RELACIONAMENTO
        faça enquanto houver LIGAÇÃO com ORIGEM = RELACIONAMENTO
        {
          Pegar ficha da LIGAÇÃO
          retirar ORIGEM
          Invalidar ficha
            Guardar ficha da LIGAÇÃO invalidada
        }
        faça enquanto houver LIGAÇÃO com DESTINO = RELACIONAMENTO
```

```
                {
                    Pegar ficha da LIGAÇÃO
                    retirar DESTINO
                    Invalidar ficha
                      Guardar ficha da  LIGAÇÃO invalidada
                }
               retirar ficha do RELACIONAMENTO
              retorne ok
          }
        senão
          {
                Mensagem "RELACIONAMENTO não encontrado"
              retorne erro exclusão
          }
fim
```

8) OBJETO LIGAÇÃO:

```
Algoritimo:  CRIAR  LIGAÇÃO

Entradas:   Origem,   Destino,TipoTraçado
Saídas:     LIGAÇÃO incluída, erro

Início
      Marcou_Origem <-   FALSO
      Marcou_Destino <-  FALSO
      faça enquanto houver ENTIDADE na tabela de ENTIDADE
            Se Origem pertence a área da ENTIDADE
              {
                 Tipo_Origem  <-  ENTIDADE
                 Nome_Origem  <-  NOME_ENT
```

```
            Marcou_Origem <- VERDADE
         }
      senão
         Se ~Marcou_Origem & Destino pertence a área da ENTIDADE
               {
                  Tipo_Destino <- ENTIDADE
                  Nome_Destino <- NOME_ENT
                  Marcou_Destino < - VERDADE
               }
Se ~(Marcou_Origem & Marcou_Destino)
      {
         faça enquanto houver RELACIONAMENTO na tabela RELACIONAMENTOS
            Se ~Marcou_Origem
               Se Origem pertence a área de RELACIONAMENTO
                     {
                        Tipo_Origem <- RELACIONAMENTO
                        Nome_Origem <- NOME_REL
                        Marcou_Origem <- VERDADE
                     }
            senão
               Se ~Marcou_Destino
                  Se Destino pertence a área de RELACIONAMENTO
                     {
                        Tipo_Destino <- RELACIONAMENTO
                        Nome_Destino <- NOME_REL
                        Marcou_Destino <- VERDADE
                     }
         }

Se ~(Marcou_Origem & Marcou_Destino)
      {
         faça enquanto houver AGREGAÇÃO na tabela AGREGAÇÃO
            Se ~Marcou_Origem
```

```
                Se Origem pertence a área de AGREGAÇÃO
                    {
                        Tipo_Origem <- AGREGAÇÃO
                        Nome_Origem <- NOME_AGR
                        Marcou_Origem <- VERDADE
                    }
                senão
                    Se ~Marcou_Destino
                        Se Destino pertence a área da AGREGAÇÃO
                            {
                                Tipo_Destino <- AGREGAÇÃO
                                Nome_Destino <- NOME_AGR
                                Marcou_Destino <- VERDADE
                            }
            }
Se ~(Marcou_Origem & Marcou_Destino)
    {
        Mensagem "LIGAÇÃO deve possuir uma ORIGEM e UM DESTINO"
        retorne erro de inclusão
    }
senão
    {
        Se ((TIPO_ORIGEM == ENTIDADE || TIPO_ORIGEM = AGREGAÇÃO) &
            (TIPO_DESTINO == RELACIONAMENTO)) ||
            ((TIPO_DESTINO == ENTIDADE || TIPO_DESTINO == AGREGAÇÃO) &
            (TIPO_ORIGEM == RELACIONAMENTO))
            {
                Selecionar da tabela de LIGAÇÃO tal que ORIGEM =
                NOME_ORIGEM &TIPO_ORIGEM = Tipo_Origem & DESTINO =
                NOME_DESTINO &TIPO_DESTINO = Tipo_Destino
                Se não encontrou LIGAÇÃO
                    {
                        Acrescentar na tabela LIGAÇÃO ficha com ORIGEM =
                        Nome_Origem, DESTINO = Nome_Destino, TIPO_ORIGEM =
                        Tipo_Origem,TIPO_DESTINO = Tipo_Destino,
```

```
                    TIPO_TRACADO = Tipo_Tracado
                }
            senão
                {
                    Mensagem "LIGAÇÃO já existente"
                    retorne erro de inclusão
                }
        }
    retorne LIGAÇÃO incluída
fim


Algoritimo: REMOVER LIGAÇÃO

Entradas: ORIGEM, DESTINO,Vértices
Saídas: LIGAÇÃO excluído, erro

Início
    Se existir LIGAÇÃO nesta Posição
    {
        Pegar ficha da LIGAÇÃO
        marcar ficha como removida
        atualizar esquema
        compactar hierarquiader
    }
    senão
    {
        Mensagem "LIGAÇÃO não encontrado nesta posição
        retorne erro exclusão
    }
fim
```

```
Algoritimo:  MUDAR TRACADO LIGAÇÃO
Entradas:  Nome_o,  Nome_d,  Tipo_Traçado
Saídas:  Instância da ligação atualizada com novo traçado

Início
    Selecionar da Tabela LIGAÇÕES tal que NOME_O = Nome_o & NOME_D = Nome_d
    Atualizar entrada com TIPO_TRACADO = Tipo_Tracado
    retorne  Ligação  atualizada
fim


Algoritimo:  REDIRECIONAR  LIGAÇÃO

Entradas:  Nome_o,  Nome_d,  Origem/Destino,  Id_objeto,  tipo_objeto  ,  x,y
Saídas:  Ligação atualizada com nova origem/destino

Início
    Selecionar da Tabela LIGAÇÃO tal que NOME_O = Nome_o & NOME_D = Nome_d
    Se  Origem/Destino  =  ORIGEM
        {
            Caso  tipo_objeto
                RELACIONAMENTO:
                   Se  TIPO_DESTINO  !=  RELACIONAMENTO
                   Atualizar entrada com ID_ORIGEM = id_objeto, TIPO_ORIGEM =
                   tipo, {Xo,Yo} = x,y

                ENTIDADE:
                AGREGAÇÃO:
                   Se  TIPO_DESTINO  !=  RELACIONAMENTO
                     {
                        Mensagem  "Não  posso  redirecionar  LIGAÇÃO"
                        retorne  erro  de  atualizacao
                     }
                   senão
```

```
                    Atualizar entrada com ID_ORIGEM = id_objeto,
                    TIPO_ORIGEM = tipo,{Xo,Yo} = x,y
            }
        senão
            {
                Caso  tipo_objeto
                    RELACIONAMENTO:
                        Se TIPO_ORIGEM != RELACIONAMENTO
                        Atualizar entrada com ID_DESTINO = id_objeto, TIPO_DESTINO
                        = tipo, {Xo,Yo} = x,y

                    ENTIDADE:
                    AGREGAÇÃO:
                        Se TIPO_ORIGEM != RELACIONAMENTO
                            {
                                Mensagem "Não posso redirecionar LIGAÇÃO"
                                retorne erro de atualizacao
                            }
                        senão
                            Atualizar entrada com ID_DESTINO= id_objeto,
                            TIPO_DESTINO = tipo, {Xo,Yo} = x,y
                }
retorne Ligação atualizada
fim
```

9) OBJETO AGREGAÇÃO:

Algoritimo: CRIAR AGREGAÇÃO

Entradas: Vértices

Saídas: AGREGAÇÃO incluída, erro

```
Início
    Obter Identificação da AGREGAÇÃO

    Selecionar da tabela AGREGAÇÃO tal que NOME_AGR = Identificação
    Se encontrou AGREGAÇÃO
        {
          Mensagem "AGREGAÇÃO já existente"
          retorne erro de inclusão
        }
    senão
        {
          Área_AGR <- Área(Vértices)
          faça enquanto houver ENTIDADE na tabela de ENTIDADE
            Se ENTIDADE pertence a Área_AGR
              Acrescentar entrada na tabela COMPÕEMAGREGA com
              NOME_AGR = Identificação,NOME = NOME_ENT, TIPO = ENTIDADE

          faça enquanto houver RELACIONAMENTO na tabela de
          RELACIONAMENTO
            Se RELACIONAMENTO  pertence a Área_AGR
              Acrescentar entrada na tabela COMPÕEMAGREGA com
              NOME_AGR = Identificação, NOME = NOME_REL, TIPO =
              RELACIONAMENTO

          faça enquanto houver AGREGAÇÃO na tabela de AGREGAÇÃO
            Se área da AGREGAÇÃO pertence a Área_AGR
              Acrescentar entrada na tabela COMPÕEMAGREGA com
          NOME_AGR = Identificação, NOME = NOME_AGR, TIPO = AGREGAÇÃO
```

```
            Acrescentar ficha na tabela AGREGAÇÃO com NOME_AGR = Identificação,
            {Xmax,Ymax} = Vérticemax, {Xmin,Ymin} = Vérticemin
            Se Vértices > 2
                  {
                     faça enquanto existir Vértices
                        Acrecentar entrada na tabela VÉRTICESAGREGAÇÃO com
                              NOME_AGR = Identificação, ORDEM = ORDEM + 1,
                              {X,Y} = Vértice
                     Atualizar tabela de AGREGAÇÃO com TEMVÉRTICE = SIM
                  }
         }
      retorne AGREGAÇÃO incluída

fim


Algoritimo: REMOVER AGREGAÇÃO

Entradas: Nome
Saídas:   AGREGAÇÃO excluída, erro

Início

      Selecionar da Tabela AGREGAÇÃo tal que NOME = Nome
      Se houver AGREGAÇÃO
         retirar ficha da Tabbel
   senão
         {
            Mensagem "AGREGAÇÃO não encontrada "
            retorne erro de exclusão
            }
fim
```

CAPÍTULO V

CONCLUSÕES

O desenvolvimento de FEGRES proporcionou a abordagem de vários aspectos a respeito de ambientes para desenvolvimento de software. Tópicos específicos sobre banco de dados, interface com o usuário e técnicas de programação precisaram ser estudados e detalhados no sentido de se assimilar uma cultura que permita desenvolver aplicativos que possam ser considerados o estado da arte em ferramentas para o engenheiro de software.

Características evolutivas do ambiente foram conseguidas através da adoção de padrões bem definidos para sua construção. A utilização de uma linguagem altamente portável, juntamente com a possibilidade de utilização de um gerenciador de banco de dados especialmente construído para suportar os objetos manuseados pelo ambiente, tornam viável sua compilação para outros tipos de equipamento, bastando uma pequena adaptação na interface caso estes novos equipamentos não possuam o Windows ou o Presentation Manager compondo o ambiente.

Dificuldades foram encontradas durante o desenvolvimento, todas elas basicamente com o sistema gerenciador de interface Windows 2.0, que devido a falta de documentação apropriada e a não disponibilidade de artigos em revistas especializadas, tornaram o desenvolvimento de FEGRES uma tarefa estafante e ao mesmo tempo estimulante. Esta falta de documentação é aparentemente provocada pela novidade que é o ambiente e a forma de programação utilizada, cujas técnicas muitas vezes assustam aos desenvolvedores de aplicações devido a quantidade de inovações existentes.

A expansão do ambiente é fortemente recomendada. Esta expansão pode ser feita acrescentando-se novas ferramentas e características adicionais ao sistema gerenciador de base de dados existentes. O porte do COPPEREL, juntamente com as extensões já existentes para ambiente Windows, através da migração de seu código originariamente escrito em linguagem

FORTRAN para linguagem C, permitirá que novos recursos sejam explorados e novas ferramentas desenvolvidas.

A criação de ferramentas adequadas que aumentem a produtividade e a qualidade das aplicações desenvolvidas para o Windows é uma tarefa que não pode ser esquecida. Apesar de já estarem disponíveis no mercado algumas aplicações que poderiam ajudar na construção de novos produtos de software, elas não se aplicam a todas as situações e constroem aplicações que não correspondem às necessidades de desempenho do ambiente.

A definição de métodos adequados para o projeto de aplicações para o Windows é necessária. Métodos estruturados não se apresentaram como bons candidatos para este tipo de desenvolvimento. A especificação da interface com o usuário não consegue ser facilmente descrita com utilização de métodos não formais ou semi formais. Apesar de possuirmos diagramas de Transição de Estados para representar as variações existentes na interface, a representação destes estados de maneira algoritimica não é tarefa fácil.

A expansão de FEGRES através da inclusão de novas ferramentas no ambiente, possibilitando uma abrangência ainda maior no processo de desenvolvimento, é viável e recomendável devido a:

- aproveitamento do conhecimento adquirido durante o desenvolvimento de FEGRES;
- custo de desenvolvimento reduzido devido a alta reutilização de código proporcionada pelo WINDOWS;
- aprovação da interface do Presentation Manager como um padrão para novas linhas de equipamentos PS/2;
- possibilidade de construir um produto de alto nível e factível de utilização.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] HAUSEN, H.L. e MULLERBURG, M., Conspectus of Software Engineering Environments, *IEEE Transactions on Software Engineering*, 1984

[2] PENEDO, M.H., Prototyping a Project Master Data Base for Software Engineering Environments, *ACM SIGPLAN*, No 1, vol. 22, 1986

[3] NUNES, D., Requisitos de Ambientes de Engenharia de Software, *Anais do I Simpósio Brasileiro de Engenharia de Software*, pp. 84-94, 1987

[4] DART,S.A., ELLISON,R.J., FEILER,P.H., and HABERMAN,A.N., Software Development Environments, *IEEE COMPUTER*, November 1987, pp. 18 -28

[5] TEITELMAN,W. e MASINER,L., The Interlisp Programming Environment, *Tutorial: Software Development Environments*, Computer Society Press, Los Alamitos, Calif., 1981, pp.73-81

[6] GOLDBERG,A., The Influence of an Object-Oriented Language on the Programming Environment, *Interactive Programming Environments*, D.R. Bastow, H.E. Shrobe, and E.Sandewall, eds, McGraw-Hill, New York, 1984, pp. 141-174

[7] FEILER,P.H. e MEDINA-MORA,R., An Incremental Programming Environment, *IEE Transactions on Software Engineering*, Sept., 1981, pp.472-482

[8] REISS,S.P., Graphical Program Development with PECAN Program Development Systems, *SIGPLAN Notes*, Proceedings ACM SIGSoft/SIGPlan Software Engineering Symposium on Pratical Software Development Environments, may 1984, pp. 30-41

[9] DOLOTTA,T.A., HAIGHT,R.C. e MASHEY, J.R., Unix Time-sharing System: The Programmer's WorkBench, *Interactive Programming Environments*, D.R.Barstow, H.E. Shrobe and E. Sandewall, eds., McGraw-Hill, New York, 1984, pp.353-3699

[10] TAYLOR, R.N. et al., Arcadia: A Software Development Environments Research Report, Techinical Report, Univ. of California, Irvine, Mar.1986

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

[12] TEICHROEW,D. e HERSHEY III,E.A., PSL/PSA: A Computer-aided Technique for Structured Documentation anda Analysis of Information Processing Systems, *IEEE Transactions on Software Engineering, Jan. 1977, pp.41-48*

[13] ROSS, D.T. e SCHOMAN Jr, K.E. , Structured Analysis for Requeriments Definition, *IEEE Transactions on Software Engineering*, Vol SE-3, No 1 [January 1977]pp 6-15

[14] GANE, C. e SARSON,T., Análise Estruturada de Sistemas, LTC, Livros Técnicos e Científicos Editora, R.J., 1984.

[15] AGUIAR, T.C., Ferramentas para Apoio à Análise Estruturada, Tese de Mestrado, IME, R.J., 1986

[16] NOGUEIRA, D.L., Ferramentas Automatizadas para Apoio ao Projeto Estruturado, Tese de Mestrado, COPPE/UFRJ, 1988

[17] YOURDON, E., MANAGING THE STRUTCTURED TECHNIQUES: Strategies for Software Development in the 1990's, Yourdon Inc 1986

[18] ROCHA, A.R.C., NOGUEIRA, D., MENEZES, M.G., BLASCHEK, J.R.S.,MATTOSO, M.L.Q. e AGUIAR, T. C., Um Conjunto de Ferramentas Automatizadas de Apoio ao Software, VI Semicro, R.J., 1986.

[19] BLASCHEK, J. R. S., Dicionário de Dados Automatizado para Análise Estruturada, Tese de Mestrado, IME, R.J., 1987

[20] VALLE, M.A.O., PRODOC: Uma Ferramenta para Produzir Documentação de Produtos de Software, Tese de Mestrado, COPPE/Sistemas, R.J., 1988

[21] MENEZES, E., ESTIME: Uma Ferramenta para Estimativa de Custo de Desenvolvimento de um Produto de Software, Tese de Mestrado, COPPE/Sistemas, R.J., 1988

[22] CAVALCANTE, S.M., Ferramenta Automatizada de Apoio ao Método DARTS, Tese de Mestrado, COPPE/UFRJ, 1988

[23] ULLOA, E.D.A., Dicionário de Dados: Uma Ferramenta Automatizada de Apoio ao Método DARTS, Tese de Mestrado, COPPE/UFRJ, 1988

[24] PAGE-JONES,M., The Pratical Guide to Strucutred Systems Design, Yourdon Press, 1980, Yourdon Inc., N.Y.

[25] MICROSOFT, Corp., Microsoft Windows Software Development Kit: Application Style Guide, Version 2.0, 1987

[26] YOURDON, E. e CONSTANTINE, L.L., Structured Design, Englewood Cliffs, Prentice Hall, 1979.

[27] MCMENAMIN, S.M. e PALMER, J.F., Essential Systems Analysis, Yourdon Inc., N.Y., 1984

[28] DEMARCO,T.,Structured Analysis anda System Specification,Yourdon Inc., N.Y., 1979

[29] ROCHA, A. R. C., Análise e Projeto Estruturado de Sistemas, Editora Campus, 1987

[30] BOEHM, B., Software Engineering as it is, *Software Engineering,* Freeman,H.; Lewis II, P.M.(ed.), Academic Press, 1980

[31] CHRISMAN, C. e BECCUE,B., Entity Relationship Models as a Tool for Data Analysis and Design, *ACM-0-89791-178-4/86/0002/0008,* pp. 8-14, 1986

[32] HOWE, D. R., Data Analysis for Data Base Design, Edward Arnold, 1983

[33] JACKSON, G. A., Relational Database Design with Microcomputer Applications, Prentice Hall, Englewood Cliffs, N. J., 1988

[34] SETZER, V.W., Projeto Lógico e Projeto Físico de Banco de Dados, V Escola de Computação, UFMG, Belo Horizonte, 1986

[35] SMITH ,J. M. e SMITH, D.C.P., Database Abstractions: Aggregation and Generalization , *ACM Transactions on Database Systems*, Vol.2, No. 2, pp. 105-133, June 1977

[36] PARNAS, D.L., On the Use of Transitions Diagrams in the Design of a User Interface for an Interactive Computer System,Proceedings 24th National ACM Conference, pp. 379-85,1969

[37] CASEY, B.E. and DASARATHY,P., Modelling the Man-Machine Interface and Validating. *Software Practice and Experience* 12[11), 557-69

[38] HEKMATPOUR,S. and INCE,D.C., Rapid Software Prototyping, *Oxford Surveys in Information Technology*, vol. 3, 37-76, Oxford University Press, 1986

[39] CODD, E.F., A Relational Model of Data for Large Shared Data Banks, *Communications of the ACM*, June, 1970

[40] CODD, E.F., Further Normalization of the Database Relational Model, *Courant Computer Science Symposio*, v.6, Database System, Englewood Cliffs, N.J., Prentice-Hall, 1972

[41] FAIRLEY, R. E., Software Engineering Concepts, McGraw-Hill Inc., 1985

[42] NEWMAN, W.N., Designing Integrated System for the Office Environment, Mc-Graw Hill Book Company, 1987

[43] PERSIANO, R.C.M. e OLIVEIRA, A.A.F., Introdução a Computação Gráfica, *V Escola de Computação*, UFMG, 1986

[44] BENNET, J.L., Collaboration of UIMS Designers and Human Factors Specialists, *Computer Graphics*, vol. 21, no 2, april 1987

[45] FOLEY, J.D., WALLACE, V.L. e CHAN,P., The Human Factors of Computer Graphics Interation Techniques, *IEEE - Computer Graphics and Application*, novembor 1986

[46] GARDINER, M.M. e CHRISTIE, B., Applying Cognitive Psychology to User-Interface Design, John Wiley & Sons, 1987

[47] SCHNEIDERMAN, B., Direct Manipulation: A Step Beyond Programming Languages, *IEEE - Computer*, august 1983

[48] HECKEL, P., The Elements of Friendly Software Design, Warner Books, 1984

[49] BERTINO, E., Design Issues in Interative User Interfaces, *Interfaces in Computing*, 3(1985) 37-53

[50] CLARK, I.A., Software Simulation as a Tool for Usable Product Design, *IBM System Journal*, vol. 20, no 3, 1981

[51] BETTS,B., Goals anda Objectives for User Interface Software, *Computer Graphics*, vol. 21, no 2, april 1987

[52] DATE, C.J., Introdução a Sistemas de Banco de Dados, Editora Campus, 1986

[53] BERSTEIN, P.A., Database Systems Support for Software Enginnering: An Extendi Approach, *ACM SIGPLAN*, 1987

[54] FOISSEAU, J., VALETTE, F.R., A Computer Aided Design Data Model: FLOREAL, *File Structures and Data Bases for CAD*, J.Encarnacao and F.L.Krause (Ed.), North-Holland Publishing Company, IFIP, 1982

[55] NASKIN, R.L. e LORIE, R.A., On Extending the Functions of a Relational Database System, *Computer Science*, RJ3182(38988), 11/11/81

[56] HITCHCOCK,P., BRAWN,A.W. e WEEDON,R., The Use of Databases for Software Enginnering, *Proceedings of the fifth British National Conference on Databases*, Canteburry, july 1986, OXBORROW (Ed.)

[57] LORIE, R.A., Issues in Databases for Design Applications, *File Strucutres and Data Bases for CAD*, IFIP, 1982

[58] POTTER,A., Software Development under Windows, *Computer Language*, Vol. 5, Number 1, January 1988, pp. 36 - 44

[59] PETZOLD, C., Programming Windows, Microsoft Press, 1988

[60] KERNIGHAN,B.W., RITCHIE, D.M., C : A Linguagem de Programação, Editora Campus, 3a edição, 1987

[61] MICROSOFT, MICROSOFT C 5.1 User's Guide, Microsoft Corp., 1988

[62] SOUZA, J.M. e MATTOSO,M.L.Q., COPPEREL-PC: A Versão do SGBD COPPEREL para Microcomputadores tipo PC, Anais do 3o Simpósio Brasileiro de Banco de Dados, março 1988

[63] MICROSOFT, Corp., Microsoft MS-DOS User's Guide: Operating System Version 3.2, 1986

[64] IBM, Corp., OS/2 Operating System, 1988

[65] YAO,P., CALSON,G. e DURANT,D., Programmer's Guide to Windows, Sybex, San Francisco, 1987

[66] MICROSOFT, Corp., Microsoft Windows Software Development Kit: Programmer's Tolls, Version 2.0, 1987

[67] TOWSEND, C. , C Programmer's Guide To Microsoft Windows 2.0, Howard W. Sams & Company, First Edition, 1988

[68] MICROSOFT, Corp., Microsoft Windows Software Development Kit: Programmer Reference, Version 2.0, 1987

[69] QUEDENS, G., Windows Virtual Machine, *PC Tech Journal*, October 1987, pp. 90 - 102

[70] MYERS,B., DONER, C., Graphics Programming under WINDOWS, SYBEX Inc., 1988

[71] SCHILDT,H. , A Presentation Manager Application Template, *Dr. Dobb's Journal*, march 1989, pp 16-27

[72] FOX, D. L., Dinamic Memory Management in C, *Byte*, August 1988

[73] DRAGANZA,M., Dynamic Link Libraries Under Windows, *Computer Language*, Vol. 6, Number 5, may 1989, pp. 59 - 74

[74] JOHNSON,M. e SOLINSKI,M., Dynamic Link Libraries Under Microsoft Windows, *Dr. Dobb's Journal*, march 1989, pp 28 - 37

[75] JAMSA, K., WINDOWS: Guia do Usuário, Osborne/McGraw-Hill, 1988

[76] CHEN, P.P., Database Design Based on Entity and Relationship, *Principles of Database Design*, Vol.1, S. Bing Yao editor, Prentice-Hal, Inc., Englewood Cliffs, N.J., 1985

[77] CRISPIM, E.M.H., Definição de Termos em Ambientes para o Desenvolvimento de Software, Relatórios Técnicos do Programa de Engenharia de Sistemas e Computação, ES-175/88, COPPE/UFRJ, Novembro de 1988

[78] ROCHA, A.R.C. e MOREIRA, J.S., TABA: Uma Estação de Trabalho Para o Engenheiro de Software,Relatórios Técnicos do Programa de Engenharia de Sistemas e Computação, ES-145/88

[79] ROCHA, A.R.C., Um Modelo para Avaliação da Qualidade de Especificações, Tese de Doutorado, PUC, R.J., 1983.

[80] TAKAHASHI, T. e HAEBERER, A.M., Proyecto ETHOS: Informe Final da Fase Preliminar, Programa Argentino-Brasileno de Investigacion y Estudos Avanzados in Informática - Enero de 1988

APÊNDICE A

MANUAL DO USUÁRIO DE FEGRES

A.1) INTRODUÇÃO

FEGRES - FErramentas GRáficas para Engenharia de Software é um ambiente que está sendo desenvolvido em linguagem C, para ambiente MS-WINDOWS, consistindo de um conjunto de ferramentas destinadas a apoiar o desenvolvimento de sistemas a partir da utilização de métodos apropriados, preocupando-se com as fases de especificação e projeto do produto. Tem como objetivo auxiliar o Analista e/ou Projetista na tarefa de construir o produto de software de maneira a economizar tempo, aumentar a qualidade do produto e reduzir os custos de desenvolvimento.

FEGRES, em sua versão atual, compõem-se de um editor de Diagrama de Fluxo de Dados, um editor de Diagramas de Entidades-Relacionamentos e um Dicionário de Dados devidamente estendido para suportar a descrição dos elementos existentes nos diagramas de Entidades-Relacionamentos.

A formação em processamento de dados e a leitura deste manual são condições indispensáveis para a perfeita utilização dos recursos disponíveis no ambiente. O uso da interface com o usuário existente é uma tarefa fácil e ao mesmo tempo empolgante e o aprendizado da forma de utilização das ferramentas é rápido e tranquilo.

Para que possamos utilizá-lo corretamente, FEGRES necessita dos seguintes recursos computacionais:

- Microcomputador compatível com IBM-PC AT
- Memória mínima de 640 Kbytes

- Placa controladora de vídeo gráfica padrão EGA

-Monitor de Vídeo para placa EGA

- Mouse

- Disco winchester de 20 Mbytes

- uma unidade de disco 5 1/4

- Impressora gráfica ou dispositivo equivalente.

Neste manual serão apresentados:

- a estrutura do ambiente;

- procedimentos de instalação;

- o modelo conceitual do usuário;

- o padrão de interface;

- descrição dos comandos existentes.

A.2) A Estrutura do Ambiente

FEGRES é um ambiente utilizado para a construção da especificação e projeto de um produto de software. Em sua versão atual possui ferramentas para os métodos da Análise Estruturada, GANE [14], e Diagramas de Entidades-Relacionamentos, CHEN [11]. Oferece todos os recursos necessários para a edição dos diagramas e construção do dicionário de dados. Utiliza-se do padrão de interface com o usuário oferecido pelo WINDOWS 2.0, MICROSOFT [25], para comunicação com o usuário.

O ambiente atual é composto de três ferramentas distintas que são o editor de diagrama de fluxo de dados, DFD, o dicionário de dados, DDADOS, e o editor de diagrama de Entidades-Relacionamentos, DER, integradas a partir de um gerenciador de base de dados baseado no modelo relacional e especialmente desenvolvido para suportar as ferramentas. Apesar deste

gerenciador ser baseado no modelo relacional, nem todas as operações possíveis sobre este modelo foram implementadas, sendo possível numa próxima versão incorporar novas características e fortalecer ainda mais as ferramentas.

Além deste gerenciador de base de dados, que funciona como uma biblioteca dinâmica, existe também um outro módulo de apoio, denominado FEGRELIB, que é responsável pelo gerenciamento do traçado dos objetos presentes em FEGRES.

A implementação de todo o ambiente foi realizada em linguagem C. A escolha desta linguagem foi feita devido a grande flexibilidade que a mesma proporciona para o desenvolvedor da aplicação , e ao mesmo tempo a possibilidade da utilização de todos os recursos disponíveis no WINDOWS.

A.3) A Instalação da Ferramenta

FEGRES é distribuída num disquete de 360 Kbytes contendo os seguintes módulos:

- DFD.EXE
- DER.EXE
- DDADOS.EXE
- FEGRELIB.EXE
- BASELIB.EXE
- CRIAESQ.EXE
- FEGRES.ESQ
- INSTALA.BAT

Cada módulo tem sua função específica e desempenha papel importante no funcionamento da ferramenta. O módulo DFD.EXE é o responsável pela edição, controle e

impressão dos diagramas de Fluxo de Dados. O módulo DER.EXE é o responsável pela edição, controle e impressão dos diagramas de Entidades-Relacionamentos. O módulo DDADOS.EXE é o responsável pela criação, edição e impressão do dicionário de dados do sistema. Os módulos FEGRELIB.EXE e BASELIB.EXE são bibliotecas dinâmicas utilizadas pelo ambiente para traçado dos objetos e controle da base de dados, respectivamente. O módulo CRIAESQ.EXE é o responsável pela construção da base de dados inicial utilizada na ferramenta e que está descrita em FEGRES.ESQ. O módulo INSTALA.BAT faz a instalação automática da ferramenta.

Para que FEGRES execute de maneira satisfatória é necessário que no equipamento estejam instalados o sistema operacional DOS versão 3.3, ou superior, e o sistema de interface WINDOWS 2.0. O procedimento de instalalação destes dois softwares é descrito nos manuais de instalação que os acompanham.

Devido ao grande volume de memória ocupado pela base de dados de FEGRES sua utilização deve ser realizada através de disco rígido, O processo de instalação se dá a partir de uma unidade de disco flexível. Insira o disquete contendo o sistema no drive A: e digite INSTALA <ENTER>. A partir deste momento começará a instalação automática das ferramentas. INSTALA criará um diretório FEGRES no disco rígido e copiará para lá todos os arquivos executáveis. Após terminada a cópia ele ativará o módulo esquema para construir a base de dados a partir do arquivo FEGRES.ESQ. Terminada esta instalação o sistema está pronto para ser utilizado.

A ativação da ferramenta deve ser feita a partir do WINDOWS. Para isto carregue o WINDOWS e escolha a partir de qual ferramenta deseja iniciar os trabalhos. Pode-se começar a trabalhar no ambiente a partir de qualquer ferramenta pois o sistema é integrado e as informações são compartilhadas entre todas elas. Recomenda-se que não se modifique o conteúdo do arquivo FEGRES.ESQ para que o ambiente não passe a funcionar incorretamente. As definições existentes neste arquivo são essenciais para a documentação da base de dados inicial de FEGRES.

A.4) O Modelo Conceitual do Usuário

A.4.1. Visão do Usuário

A atividade do usuário na utilização da ferramenta é no desenvolvimento de um PROJETO.

O período contínuo de utilização da ferramenta por parte do usuário é denominado uma SESSÃO. Em uma SESSÃO o usuário tem acesso a uma ou mais UNIDADES que contêm PASTAS,que por sua vez podem conter PROJETOS e/ou PASTAS armazenadas.

Uma UNIDADE representa um dispositivo lógico de recuperação/armazenamento de informação, que pode ser por exemplo uma unidade de disco. O usuário no início da utilização da ferramenta deve indicar qual UNIDADE ele desejará utilizar.

Uma PASTA representa um diretório ou uma área de trabalho num disco. A escolha do PROJETO se faz a partir de sua identificação na PASTA que está ativa no momento. Esta PASTA pode conter outras PASTAS , PROJETOS e informações pertencentes ao usuário.

Um PROJETO é composto de DIAGRAMAS, DICIONÁRIO DE DADOS e DESCRIÇÃO DA LÓGICA DE PROCESSOS.

Feita a escolha do PROJETO, a ferramenta apresentará ao usuário o nível de contexto do DIAGRAMA DE FLUXO DE DADOS pertencente ao PROJETO em questão. Caso o PROJETO seja novo, o nível de contexto estará vazio.

Tendo escolhido o PROJETO, o usuário está com os DIAGRAMAS pertencentes ao PROJETO em suas mãos. Ele pode trabalhar com eles como se estivesse manuseando uma folha

de papel, com a ressalva de que apenas uma parte de cada vez dessa folha pode ser visualizada por ele.

Toda informação acrescida pelo usuário a respeito de elementos dos DIAGRAMAS é automaticamente guardada e gerenciada pelo DICIONÁRIO DE DADOS.

Os DIAGRAMAS do PROJETO são compostos pelas seguintes representações:

.DIAGRAMA DE FLUXO DE DADOS (DFD)

Composto dos símbolos:

1) ENTIDADE EXTERNA;
2) PROCESSO;
3) FLUXO DE DADOS;
4) DEPÓSITO DE DADOS.

A forma de construção de um DFD faz com que cada nível apresentado seja um detalhamento maior do nível anterior. Isto provoca representações diferenciadas ao longo da especificação. Se o usuário necessitar detalhar um PROCESSO isto causará o surgimento de uma explosão deste PROCESO, que estará num nível inferior ao do PROCESSO escolhido e que é por si um novo DFD. A necessidade do usuário detalhar o modelo de dados do sistema, a partir de um DEPÓSITO DE DADOS qualquer provocará a apresentação de um Diagrama de Entidades-Relacionamentos.

.DIAGRAMA DE ENTIDADES-RELACIONAMENTOS (DER)

Composto dos símbolos:

1) ENTIDADE;
2) RELACIONAMENTO;

3) LIGAÇÃO;

4) AGREGAÇÃO.

Os DER's representam o relacionamento entre as estruturas de informações contidas no sistema modelado pelo usuário.

Um modelo de Entidades-Relacionamentos representado para o sistema é único, onde seus elementos estão armazenados nos DEPÓSITOS DE DADOS existentes no DFD.

.DICIONÁRIO DE DADOS (DDADOS)

O DDADOS é o fichário no qual o usuário guardará toda a informação necessária a compreenssão e documentação do PROJETO.

Todas as entradas realizadas pelo usuário a partir dos diagramas são armazenadas automaticamente no DDADOS e gerenciadas por ele.

O DDADOS é composto de FICHAS para a descrição de:

1) ENTIDADES EXTERNAS;

2) PROCESSOS;

3) FLUXOS DE DADOS;

4) DEPÓSITO DE DADOS;

5) ESTRUTURA DE DADOS;

6) ELEMENTO DE DADOS;

7) ÍTENS DE GLOSSÁRIO;

8) ENTIDADES;

9) RELACIONAMENTOS;

10) AGREGAÇÕES;

11) LIGAÇÕES.

A.4.2. Características dos Objetos

A.4.2.1 SESSÃO

Uma SESSÃO representa o ato do usuário utilizar a ferramenta, ou melhor, o período contínuo de utilização da ferramenta.

Caracteriza-se por:

1) Data da sessão;

2) Hora de início da sessão;

3) Hora de término da sessão;

4) Nome do usuário.

O objeto SESSÃO consegue executar as seguintes tarefas:

1) ABRIR: abrir uma sessão representa o ato do usuário inicializar a ferramenta, colocá-la em funcionamento. Para que ocorra uma abertura de sessão é necessário que o usuário invoque a ferramenta e forneça seu nome;

2) FECHAR: fechar uma sessão representa o término de serviço por parte do usuário. Ao aplicar esta função ele está dizendo para a ferramenta que já terminou sua tarefa, guardou seus PROJETOS e PASTAS e não deseja mais continuar utilizando-a;

3) TROCAR_NOME_USUÁRIO: esta função tem a finalidade de facilitar ao usuário a possibilidade de corrigir seu nome caso ocorra algum erro no fornecimento desta informação na abertura da sessão. Esta correção só é possível de ser realizada uma vez;

A.4.2.2 UNIDADE

Uma UNIDADE representa um dispositivo de armazenamento/recuperação de informação. Este dispositivo pode ser um acionador de disquete, uma unidade de disco rígido ou então algum outro dispositivo de armazenamento pertencente ao conjunto de periféricos conectados ao computador em que esteja sendo executada a ferramenta. Uma UNIDADE pode conter PASTAS e /ou outras informações que não interessem para a ferramenta. A ferramenta considera como UNIDADE ativa aquela da qual foi aberta a SESSÃO.

Caracteriza-se por :

1) Nome da Unidade.

Uma UNIDADE executa a seguinte tarefa:

1) ATIVAR_UNIDADE: esta função serve para o usuário ordenar a ferramenta que mude a UNIDADE ativa. Para isto o usuário deve fornecer o nome da nova UNIDADE desejada. A execução desta função fará com que todas as PASTAS utilizadas a partir desse momento sejam procuradas nesta nova unidade.

A.4.2.3 PASTA

Uma PASTA representa um local onde estão contidos PROJETOS e/ou outras PASTAS. O número de PROJETOS e PASTAS que podem estar contidos numa determinada PASTA depende exclusivamente do tamanho desta PASTA. Existe uma associação direta entre uma PASTA e um diretório ou área de trabalho em disco.

Caracteriza-se por:

1) Nome da Pasta.

São as seguintes as funções que uma PASTA pode executar:

1) ESCOLHER_PASTA: esta função serve para o usuário dizer que PASTA ele deseja utilizar. Para isto ele deve fornecer o nome da PASTA. A ferramenta procurará a PASTA na UNIDADE que o usuário tiver escolhido anteriormente. A execução desta função fará com que a PASTA corrente seja desativada e passada a considerar a nova PASTA escolhida;

2) IMPRIMIR_PASTA: esta função fará com que as informações contidas numa dada PASTA sejam impressas para que o usuário possa guardá-las;

3) CRIAR_PASTA: esta função permite ao usuário criar uma nova pasta na PASTA que estiver ativa. Para que isso aconteça é necessário que o usuário informe o nome da nova PASTA;

4) DUPLICAR_PASTA: esta função permite ao usuário fazer uma cópia da PASTA. Esta cópia é de grande utilidade pois proporciona ao usuário uma forma de aumentar a segurança e seus projetos. Para que isto ocorra é necessário que o usuário forneça o nome da UNIDADE para qual a PASTA será copiada. Será copiado todo o conteúdo da PASTA que estiver ativa no momento, ou seja, aquela que foi anteriormente escolhida pelo usuário;

5) RETIRAR_PASTA: esta função permite ao usuário retirar uma PASTA de uma UNIDADE, juntamente com toda informação contida dentro dela. A perda de informações que não pertençam à ferramenta mas que estejam armazenadas nesta PASTA ocorrerá. Para que esta função execute é necessário que o usuário confirme satisfatoriamente após selecioná-la.

A.4.2.4 PROJETO

Um PROJETO representa o projeto que o usuário está desenvolvendo, ou seja, toda a especificação do sistema.

Caracteriza-se por:

1) Código do Projeto;

2) Nome do Projeto;

3) Orgão Responsável;

4) Nome do Gerente;

5) Número da Versão;

6) Data de Início do Projeto;

7) Data da Última atualização;

8) Autor da Última Atualização;

9) Hora da Última Atualização.

O objeto PROJETO consegue executar as seguintes tarefas:

1) CRIAR_PROJETO: esta função permite ao usuário criar um novo PROJETO na PASTA que estiver ativa no momento. Para que isto ocorra é necessário que o usuário forneça o nome do novo projeto;

2) ESCOLHER_PROJETO: esta função representa o ato do usuário escolher um certo PROJETO na PASTA que estiver ativa no momento. Esta escolha é feita a partir da indicação do nome do projeto;

3) ALTERAR_ATRIBUTOS_PROJETO: esta função permite ao usuário alterar os atributos de um PROJETO que esteja ativo no momento. Para isto ele deve informar o atributo desejado e seu novo conteúdo. O efeito causado pela troca do nome do projeto será a renomeação do mesmo na PASTA em que se localiza;

4) RETIRAR_PROJETO: esta função fará com que um projeto seja retiradao da PASTA. Para execução desta função é necessário que o usuário informe o nome do PROJETO;

5) DUPLICAR_PROJETO: esta função proporciona ao usuário a facilidade de poder duplicar um determinado projeto, ou seja, copiá-lo para uma nova PASTA. Para isso é preciso que o PROJETO já tenha sido escolhido pelo usuário. Esta função, para sua execução, necessita saber qual a UNIDADE e a PASTA que conterão a cópia do PROJETO;

6) FECHAR_PROJETO: esta função indica para a ferramenta que o usuário terminou a utilização de um determinado PROJETO. A aplicação desta função fará com que todas as informações referentes ao PROJETO sejam guardadas em sua PASTA.

A.4.2.5 DIAGRAMAS

DIAGRAMAS representam os diagramas existentes na especificação de um sistema, e definidos conforme os métodos utilizados.

Caracterizam-se por:

1) DIAGRAMA DE FLUXO DE DADOS (DFD);

2) DIAGRAMA DE ENTIDADES-RELACIONAMENTOS (DER).

As funções que o usuário pode aplicar sobre este objeto são:

1) EDITAR_DIAGRAMA: esta função permite ao usuário a edição dos diagrmas de seu PROJETO. Através desta função ele pode fazer inclusões , alterações e deleções de elementos. Todas as alterações decorrentes de uma edição serão automaticamente armazenadas pelo DDADOS. Isto se torna necessário para que possamos manter a documentação a mais consistente possível;

2) APAGAR_DIAGRAMAS: esta função permite ao usuário apagar todos os DIAGRAMAS existentes em seu projeto de uma vez. Para que a consistência da especificação seja mantida

todas as FICHAS referentes ao DIAGRAMA existente no DDADOS devem ser também removidas. A ferramenta automaticamente já executa a tarefa de remover as fichas;

3) IMPRIMIR_DIAGRAMAS: esta função permite ao usuário imprimir todos os DIAGRAMAS existentes em seu projeto. A impressão será feita utilizando a metáfora do WYSIWYG ("what you see is what you get");

4) GUARDAR_DIAGRAMA: esta função permite ao usuário forçar a ferramenta a guardar alterações feitas nos DIAGRAMAS de seu PROJETO na PASTA correspondente durante uma SESSÃO sem a necessidade de encerrar seu trabalho;

5) MOSTRAR_DIAGRAMAS: esta função permite ao usuário visualizar todos os DIAGRAMAS de um PROJETO sem possibilitar edição. Sua utilização é aconselhada para consultas rápidas e seguras do PROJETO e para se evitar modificar algum dado sem necessidade.

A.4.2.6 DIAGRAMA DE FLUXO DE DADOS

Representa a descrição funcional do sistema, mostrando as possibilidades de fluxo de informação e suas transformações ao longo do sistema.

Caracteriza-se por:

1) Tela de Representação;

2) Nível;

3) Código do Projeto.

Um DFD é composto dos seguintes elementos:

1) ENTIDADE EXTERNA: representa qualquer entidade lógica que não faz parte dos limites do sistema mas que interage com ele, recebendo dados do sistema ou fornecendo dados para

o sistema.

Caracteriza-se por:

1) Identificação da Entidade Externa;

2) Tela;

3) Posição no DFD;

4) Instância.

2) PROCESSO: representa o elemento responsável pela transformação dos dados.

Caracteriza-se por:

1) Identificação do Processo;

2) Tela;

3) Posição;

4) Instância.

3) DEPÓSITO DE DADOS: representam os locais onde os dados serão armazenados.

Caracteriza-se por:

1) Identificação do Depósito de Dados;

2) Tela;

3) Posição;

4) Instância.

4) FLUXO DE DADOS: representam os caminhos por onde a informação pode fluir no sistema.

Caracteriza-se por:

1) Nome do Fluxo de Dados;

2) Origem;

3) Destino;

4) Vértice da Origem;

5) Vértice do Destino;

6) Tela;

7) Instância;

8) Tipo do Traçado.

O objeto DIAGRAMA DE FLUXO DE DADOS, juntamente com seus elementos, pode executar as seguintes tarefas:

1) INSERIR_ELEMENTO: esta função permite ao usuário inserir um elemento no DFD. Para que isto ocorra é necessário que o usuário indique a posição e qual o tipo de elemento desejado. Caso esteja inserindo um Fluxo de Dados é necessário que sejam marcados a origem e o destino do fluxo, juntamente com o tipo de traçado a ser associado ao mesmo. Após a escolha uma FICHA é aberta pelo DDADOS para que informações complementares de identificação do elemento sejam obtidas e armazenadas no DDADOS, caso esta ficha ainda não tenha sido armazenada;

2) MOVER_ELEMENTO: esta função permite ao usuário mudar a posição de um elemento no DFD. Para que isto ocorra o usuário deve selecionar o elemento correspondente. Um Fluxo de Dados só pode ser movido através da movimentação da sua origem ou de seu destino;

3) REMOVER_ELEMENTO: esta função permite ao usuário retirar um elemento do DFD. Toda informação agregada a este elemento será retirada da especificação, caso não ocorra mais nenhuma repetição deste elemento;

4) EXPLODIR_PROCESSO: esta função permite que o usuário represente com mais detalhe a descrição funcional de um determinado processo a partir da criação de uma explosão para o processo. Esta explosão é um novo DFD e o processo explodido se torna o pai deste DFD. Para a execução desta função é necessário que o usuário indique qual processo que deseja

explodir. Todos os elementos associados a este processo através de Fluxos de Dados que entram e saem são automaticamente carregadas para o desenho da explosão;

5) RETORNAR_PROCESSO_PAI: esta função permite ao usuário retornar ao DFD que contém o processo pai deste DFD. O usuário não precisa informar mais nada para essa função;

6) DETALHAR_DEPÓSITO_DE_DADOS: esta função permite que o usuário acesse a partir do DFD o modelo de dados do sistema representado através de um Diagrama de Entidades-Relacionamentos. Para que esta função execute é necessário que o usuário indique a partir de qual depósito de dados deseja chegar ao DER. Os elementos do DER que estão armazenados neste depósito de dados serão apresentados de forma destacada no diagrama;

7) REDIRECIONAR_FLUXO: esta função permite ao usuário modificar a origem ou o destino de um Fluxo de Dados;

8) MUDAR_TIPO_TRAÇADO: esta função permite ao usuário modificar o tipo de traçado de um fluxo de dados.

A.4.2.7 DIAGRAMAS DE ENTIDADES- RELACIONAMENTOS (DER)

Diagramas de Entidades-Relacionamentos representam o modelo de dados do sistema, ou seja, mostram o relacionamento entre as estruturas de informação contidas no sistema construído pelo usuário.

Caracterizam-se por:

1) Código do Projeto.

Um DER é composto dos seguintes elementos:

1) ENTIDADE: é uma representação abstrata de um "objeto" do mundo real, manuseado

pelo sistema.

Caracteriza-se por:

1) Nome da Entidade;

2) Posição;

3) Atributos Determinantes;

4) Atributos Genéricos.

2) RELACIONAMENTO: representa as associações entre as entidades.

Caracteriza-se por:

1) Nome do Relacionamento;

2) Posição;

3) Atributos Determinantes;

4) Atributos Genéricos.

3) LIGAÇÃO: indica a relação da ENTIDADE num RELACIONAMENTO.

Caracteriza-se por:

1) Nome da Origem;

2) Nome do Destino;

3) Vértice da Origem;

4) Vértice do Destino;

5) Totalidade;

6) Cardinalidade;

7) Tipo do Traçado.

4) AGREGAÇÃO: representam a reunião de ENTIDADES e RELACIONAMENTOS formando um elemento comum.

Caracterizam-se por:

1) Nome da Agregação;

2) Vértices;

3) Elementos Constituintes.

As seguintes funções podem ser executadas pelo objeto DER:

1) INSERIR_ELEMENTO: esta função permite ao usuário inserir um elemento no DER. Para que isto ocorra é necessário que o usuário indique a posição e qual o tipo de elemento desejado. Caso estaja inserindo uma Ligação é necessário que sejam marcados a origem e o destino da Ligação, juntamente com o tipo de traçado a ser associado a mesma. Após a escolha uma FICHA é aberta pelo DDADOS para que informações complementares de identificação do elemento sejam obtidas e armazenadas no DDADOS, caso ainda não exista;

2) MOVER_ELEMENTO: esta função permite ao usuário mudar a posição de um elemento no DER. Para que isto ocorra o usuário deve selecionar o elemento correspondente. Uma Ligação só pode ser movida através da movimentação da sua origem ou seu destino;

3) REMOVER_ELEMENTO: esta função permite ao usuário retirar um elemernto do DER. Toda informação pertencente a este elemento será retirada da especificação;

4) REDIRECIONAR_LIGAÇÃO: esta função permite ao usuário modificar a origem ou o destino de uma Ligação;

5) MUDAR_TIPO_TRAÇADO: esta função permite ao usuário modificar o tipo de traçado de uma Ligação;

6) DETALHAR_MODELO_LÓGICO: esta função permite ao usuário ativar o Diagrama de

Fluxo de Dados associado para verificar a especificação a nível de diagramas lógicos;

7) DETALHAR_MODELO_FUNCIONAL: esta função permite ao usuário ativar o Diagrama de Transição de Estados para veriicar a especificação a nível de diagramas funcionais;

A.4.2.8 DICIONÁRIO DE DADOS

Representa o dicionário de dados do sistema, conforme definido no método proposto.

Caracteriza-se por:

1) Código do Projeto;

2) FICHAS.

O objeto Dicionário de Dados pode executar as seguintes funções:

1) EDITAR_DICIONÁRIO: esta função permite ao usuário a edição do Dicionário de Dados de seu projeto. Através desta função ele pode fazer alterações nas descrições contidas nas FICHAS que descrevem os elementos pertencentes ao sistema;

2) IMPRIMIR_DICIONÁRIO: esta função permite ao usuário imprimir todas as FICHAS contidas no Dicionário de Dados;

3) IMPRIMIR_REFERÊNCIAS_CRUZADAS: esta função permite que o usuário imprima as relações entre os vários objetos descritos nas FICHAS existentes no DICIONÁRIO DE DADOS;

4) GUARDAR_DICIONÁRIO: esta função permite ao usuário forçar a ferramenta a guardar quaisquer alterações feitas no Dicionário de Dados de seu Projeto na Pasta correspondente durante a realização de um a SESSÃO sem a necessidade de encerrar o trabalho;

5) APAGAR_DICIONÁRIO: esta função permite que o usuário limpe todas as FICHAS contidas no Dicionário de Dados. Esta limpeza não retira a Ficha do Dicionário;

6) MOSTRAR_DICIONÁRIO: esta função permite que o usuário visualize todo o Dicionário de Dados sem entrar no modo de edição. A finalidade desta função é aumentar a segurança da ferramenta, evitando alguma alteração acidental na consulta de uma Ficha.

A.4.2.9 FICHAS

Representam as fichas de descrição dos elementos pertencentes a um PROJETO armazenadas no DICIONÁRIO DE DADOS.

O objeto FICHAS se compõem dos seguintes elementos:

1) FICHA_ENTIDADE_EXTERNA: é a ficha para descrição da Entidade Externa no Dicionário de Dados.

Caracteriza-se por:

1) Identificação da Entidade Externa;
2) Nome da Entidade Externa;
3) Linguagem;
4) Hardware;
5) Pessoa de PD;
6) Fone de PD;
7) Setor;
8) Pessoa da Org;
9) Cargo;
10) Fone da Org.

2) FICHA_PROCESSO: é a ficha para descrição de um Processo no Dicionário de Dados.

Caracteriza-se por:

1) Identificação do Processo;

2) Nome do Processo;

3) Descrição;

4) Resumo Lógico;

5) Referências Físicas;

6) Detalhes da Lógica.

3) FICHA_FLUXO_DADOS: é a ficha para descrição do Fluxo de Dados no Dicionário de Dados.

Caracteriza-se por:

1) Nome do Fluxo;

2) Descrição;

3) Descrição Ampliada;

4) Estruturas/Elementos relacionados;

5) Volume.

4) FICHA_DEPÓSITO_DADOS: é a ficha para descrição de um Depósito de Dados no Dicionário de Dados.

Caracteriza-se por:

1) Identificação do Depósito;

2) Nome do Depósito;

3) Estruturas/Elementos relacionados;

4) Acesso Imediato;

5) Organização Física.

5) FICHA_ESTRUTURA_DADOS: é a ficha para descrição de uma Estrutura de Dados no Dicionário de Dados.

Caracteriza-se por:

1) Nome da Estrutura;

2) Descrição;

3) Estruturas/Elementos de Dados relacionados;

4) Volume.

6) FICHA_ELEMENTO_DADOS: é a ficha para descrição de um Elemento de Dados no Dicionário de Dados.

Caracteriza-se por:

1) Nome do elemento;

2) Descrição;

3) Tipo do Elemento;

4) Valores/Significados;

5) Domínio;

6) Pseudônimos;

7) Outras Referências.

7) FICHA_GLOSSÁRIO: é a ficha de descrição para um Ítem de Glossário no Dicionário de Dados.

Caracteriza-se por:

1) Nome do elemento;

2) Descrição;

3) Tipo do Elemento;

4) Valores/Significados;

5) Domínio;

6) Pseudônimos;

7) Outras Referências.

8) FICHA_ENTIDADE: é a ficha de descrição de uma Entidade no Dicionário de Dados.

Caracteriza-se por:

1) Nome da Entidade;

2) Atributos Determinantes;

3) Atributos Genéricos.

9) FICHA_RELACIONAMENTO: é a ficha de descrição do Relacionamento no Dicionário de Dados.

Caracteriza-se por:

1) Nome do Relacionamento;

2) Atributos Determinantes;

3) Atributos Genéricos.

10) FICHA_AGREGAÇÃO: é a ficha de descrição de uma Agregação no Dicionário de Dados.

Caracteriza-se por:

1) Nome da Agregação;

2) Elementos Componentes.

11) FICHA_LIGAÇÃO: é a ficha de descrição da Ligação existente no Dicionário de Dados.

Caracteriza-se por:

1) Nome da Origem;

2) Nome do Destino;

3) Totalidade;

4) Cardinalidade.

O objeto FICHAS pode executar as seguintes tarefas:

1) IMPRIMIR_FICHA: esta função permite ao usuário imprimir uma FICHA de um elemento anteriormente selecionado ;

2) IMPRIMIR_FICHAS: esta função permite que o usuário imprima as FICHAS correspondentes a um determinado tipo de elemento. Para isto ocorrer é necessário que ele escolha qual tipo de elemento deseja imprimir;

3) ALTERAR_FICHA: esta função permite ao usuário alterar informações contidas numa determinada ficha. Para isto basta que ele forneça o tipo de elemento desejado juntamente com a identificação do elemento. Esta função não permite que o usuário mude a identificação e o nome do elemento;

4) LIMPAR_FICHA: esta função permite que o usuário apague as informações contidas numa ficha. Para isto basta que forneça o tipo de elemento desejado e a identificação do elemento. Esta função não permite que o usuário retire a ficha do Dicionário de Dados;

5) ORDENAR_FICHAS: esta função faz com que um determinao tipo de ficha seja ordenado alfabeticamente. Esta função se torna muito útil quando o usuário necessita imprimir este conjunto de Fichas para arquivamento.

A.5) O Padrão da Interface com o usuário

A.5.1) Características do WINDOWS 2.0

A.5.1.1) Descrição do Produto

Windows, MICROSOFT [25], é um ambiente operacional que executa em equipamentos PC compatíveis sobre sistema operacional MS-DOS, versão 2.0 ou superior, MICROSOFT [63]. Foi anunciado pela Microsoft Corporation em novembro de 1983 e efetivamente lançado no mercado em novembro de 1985. A versão 2.0 do Windows incorpora em seu núcleo mudanças consideráveis em relação a versões anteriores para que possa manter consistência com a interface com o usuário do OS/2 Presentation Manager, ambiente de comunicação do novo sistema operacional OS/2, IBM [64], lançado em novembro de 1987.

A utilização do WINDOWS faz com que passemos a trabalhar com um ambiente extremamente poderoso, com disponibilidade de recursos avançados e de fácil utilização por parte do usuário final. A capacidade de processamento é aumentada devido ao WINDOWS simular e explorar características de ambientes multi-tarefas, onde podemos ter mais de uma aplicação compartilhando os recursos do computador.

A.5.1.2) A Interface com o Usuário do WINDOWS 2.0

Microsoft Windows pode ser definido como um ambiente de programação especial que provê uma extensão ao DOS. Proporciona ao programador da aplicação a possibilidade de utilizar três capacidades básicas: uma interface com o usuário orientada a gráficos, um ambiente multi-tarefa e uma independência de hardware.

A interface com o usuário do Windows utiliza-se de figuras para simbolizar os comandos do sistema, ações e os vários dispositivos existentes na sua interface visual. Devido ao aspecto visual

da interface, largamente utilizado e enfatizado pelo ambiente ,onde objetos podem ser manipulados livremente e de forma transparente ao usuário, o termo interface visual é melhor empregado do que interface gráfica.

Um objeto gráfico para o Windows é uma coleção de dados que pode ser manipulado como um única entidade e apresentado ao usuário como parte da interface visual, YAO [65].O coração desta interface é o GDI (" Graphics Device Interface") que na sua essência compõem-se de uma biblioteca de rotinas especialmente construídas para tratar a apresentação visual dos objetos componentes da interface. Esta biblioteca é acessível tanto pela aplicação como pelo próprio Windows.

Aplicações, para aproveitarem de maneira satisfatória os recursos do ambiente, devem ser especialmente construídas para ele através da utilização de ferramentas próprias fornecidas pela Microsoft e denominadas Kit de desenvolvimento.(" Windows Development Kit"), MICROSOFT [66] . O uso destas ferramentas para construção da aplicação garantem a perfeita utilização dos recursos disponíveis no ambiente além de manterem a consistência de construção e apresentação entre as aplicações.

O desenvolvimento de aplicações para o Windows não é tarefa fácil para programadores que não estejam em contato com técnicas e conceitos modernos de programação, tendo em vista as várias inovações incorporadas no ambiente. Utiliza-se em grande escala a linguagem C, com algumas incursões pelo Assembler, o que restringe ainda mais o universo de profissionais capazes de desenvolver aplicações num nível de qualidade aceitável.

Dois aspectos precisam ser considerados quando tratamos do desenvolvimento de aplicações para o Windows: o que diz respeito ao usuário e o que diz respeito ao desenvolvedor da aplicação. A utilização da interface pelo usuário é o ponto agradável do Windows. Ele fica extremamente a vontade no seu uso , pois a mesma não impõem restrições de utilização, utilizando de recursos próximos a forma de pensar do ser humano. Ao contrário, no que diz

respeito ao desenvolvedor, este tem que se preocupar em projetar a interface de sua aplicação no sentido de utilizar de maneira eficiente todos os recursos disponíveis e necessários, e também, de utilizar técnicas não convencionais de desenvolvimento para conseguir aproveitar realmente todos os recursos do ambiente.

A.5.1.3) Objetos Disponíveis na Interface Visual

1) A JANELA:

O principal objeto gráfico do GDI, e que forma a essência do ambiente, é a janela. O papel da janela na interface visual é servir como uma abertura na tela de apresentação que permita ao usuário visualizar a aplicação que está associada a ela, transformando-se num canal de comunicação entre o usuário e a aplicação .

A tela pode ser dividida em várias janelas, cada uma associada a uma aplicação específica, ou então, associada a várias instâncias da mesma aplicação, onde o usuário consegue receber ou enviar informações para a janela que está ativa no momento. Não importa o número de janelas existentes na tela, nem o tamanho de cada uma delas, o Windows gerenciará este espaço de maneira eficiente e consistente e é através dele que permitirá ao usuário utilizar todos os recursos disponíveis no sistema.

A distribuição do espaço da tela entre as várias janelas existentes é realizada pelo próprio Windows. Em versões anteriores , janelas eram alocadas através do compartilhamento da área de apresentação. A disposição das janelas na tela era feita utilizando-se janelas justapostas ("tiled windows"). Assim, quando uma nova janela era colocada na tela, todas as outras dimimuiam seu tamanho para poder alocar espaço para esta nova janela. A versão 2.0 do Windows utiliza-se de janelas sobrepostas ("overlapped windows"). As janelas são alocadas se sobrepondo umas as outras permitindo com isso um melhor aproveitamento do espaço de apresentação e melhorando muito o aspecto da

interface. A figura (A.1) apresenta a aparência de janelas sobrepostas.

Figura A.1 - Aparência de Janelas Sobrepostas

O Windows divide uma janela em seis áreas principais, que podem estar presentes nas janelas apresentadas, com as quais o usuário pode realizar controles sobre sua aplicação. Identificam-se as seguintes áreas:

. A caixa de cardápio do sistema e o cardápio do sistema que permitem ao usuário controlar aspectos de apresentação da janela, como movimento, tamanho e existência, situada no canto esquerdo superior;

. A barra de título que contém o nome da janela e está localizada no topo central da janela;

. A caixa de dimensionamento que é utilizada pelo usuário para modificar o tamanho da janela através do mouse, localizada no canto direito superior;

. A barra de cardápios da aplicação, que é usada para construir cardápios e executar seleções e está localizada abaixo da barra de título;

. As barras de rolamento horizontal e vertical que servem para rolar o que está sendo apresentado na janela e se localizam ao longo da dimensão horizontal inferior e dimensão vertical a direita, respectivamente, e;

. A área da aplicação, conhecida como área do cliente ("client area"), que é a área de trabalho da aplicação onde serão apresentadas as informações geradas pela aplicação. Somente esta área a aplicação pode desenhar e controlar diretamente.

Na figura (A.2) podemos observar em maior detalhe como o Windows compõe uma janela. A existência destes objetos que não pertencem a área da aplicação está diretamente ligada à forma como a janela foi definida e criada na aplicação. Janelas pertencem a uma determinada classe, TOWSEND[67], YAO[65], MICROSOFT [68], e herdam características definidas nesta classe. Quando a aplicação cria uma janela de uma determindada classe define como esta janela será apresentada na tela. Apesar de terem sido criados pela aplicação, estes objetos estão sob controle direto do Windows, garantindo com isto que eles reajam de maneira uniforme para as entradas do mouse e do teclado.

Janelas podem ser de três tipos: Sobrepostas ("Overlapped"), Pop-Up e filhas ("Child"). O tipo é definido no momento da criação da janela pela aplicação e executarão ações específicas conforme as mensagens que a mesma possa responder. Estas mensagens são enviadas pelo Windows, ou por uma aplicação, e podem ser visualizadas como eventos que provocam uma mudança de estado da aplicação. Todas as janelas, indenpendente de estar ativa ou não, estão recebendo mensagens a todo instante e respondendo-as de acordo com seu estado na tela.

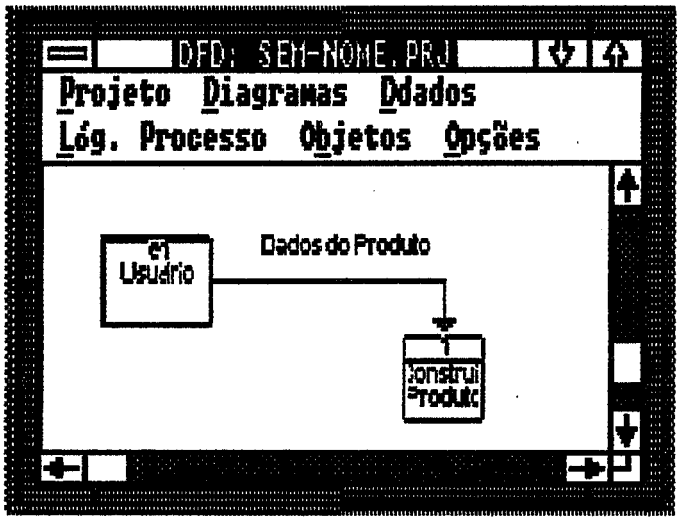

Figura A.2 - Componentes de uma Janela

2) ÍCONES

Um ícone é um pequeno símbolo utilizado para chamar a atenção do usuário para um objeto que seja importante para a aplicação e que cause um reconhecimento rápido por parte do usuário.

Toda aplicação deve possuir um ícone associado que a identifique de maneira única. O uso de ícones é largamente utilizado pelo Windows para representar aplicações que estão carregadas mas suas janelas associadas possuem dimensões mínimas. Utiliza-se também ícones para destacar para o usuário ações que devam ser realizadas imediatamente ou realçar mensagens importantes do sistema. A figura (A.3) apresenta ícones de aplicações e ícones usados para algum tipo de advertência.

Ícones são armazenados como um recurso da aplicação. Apesar da aplicação poder livremente criar e modificar ícones, o Windows possui alguns armazenados e prontos para uso. Estes ícones armazenados pelo Windows são componentes dos chamados objetos de estoque ("Stock Objects"). Estes objetos estão disponíveis para qualquer aplicação no

tempo em que elas necessitarem.

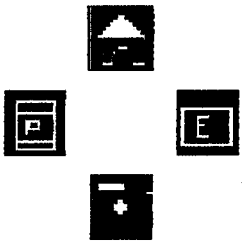

Figura A.3 - Ícones

3) CURSORES

Cursores são dispositivos fornecidos pelo Windows através dos quais o usuário pode indicar uma localização de interesse dentro da janela .A aparência do cursor deve ser usada para comunicar o tipo de ação que deveria ocorrer se fosse apertado o botão do mouse enquanto apontado por este cursor. Esta aparência pode ser trocada dinamicamente durante a aplicação.

Um cursor pode ser associado a uma janela no momento de definição de sua classe. A partir daí todas as janelas daquela classe passarão a possuir este cursor. Uma aplicação pode, também, durante sua execução, modificar, temporariamente, ou definitivamente, o tipo de cursor associado a sua janela aumentando assim a flexibilidade da interface.

É de responsabilidade do Windows controlar a aparência do cursor que esteja sobre uma determinada janela. O fato de movimentarmos o mouse pela tela, passando o cursor pelas diversas janelas que por ventura existam em exibição naquele instante, provocará a mudança da aparência do cursor para cada janela apresentada, caso elas utilizem cursores diferentes.

4) Caixas de Mensagem e Caixas de Diálogo

Caixas de Mensagem são janelas "Pop-Up" que possuem um ícone e uma mensagem para o usuário conforme pode ser observado na figura (A.4) . São fáceis de programar e largamente utilizadas para chamar a atenção imediata do usuário, YAO [65]. Provocam uma parada momentânea da aplicação até que o usuário responda que recebeu a mensagem e retorne alguma informação necessária para prosseguimento de prcoessamento. Devido a isto este tipo de diálogo é conhecido como modal. A criação de caixas de mensagem pela aplicação é feita através da utilização de rotina específica do GDI. Para esta rotina o usuário deve prover o texto da mensagem, o título da caixa de mensagem e, se quizer, um ícone a ser desenhado junto com a mensagem.

Caixas de Diálogo também são janelas "Pop-Up". São similares às caixas de mensagens mas se prestam muito bem a receber dados de entrada do usuário. Na realidade são a principal forma de se obter os dados de entrada do usuário. A criação de uma caixa de diálogo se faz através de descrição textual dos elementos que a compõem num arquivo apropriado e da utlização de rotinas internas do GDI. Caixas de diálogo modais são aquelas que não permitem que a aplicação continue seu processamento antes que o usuário termine a entrada de informação. Caixas de diálogo não modais são aquelas que podem estar conversando com usuário concorrentemente com a aplicação. A figura (A.5) apresenta uma caixa de diálogo.

Figura A.4 - Caixa de Mensagem

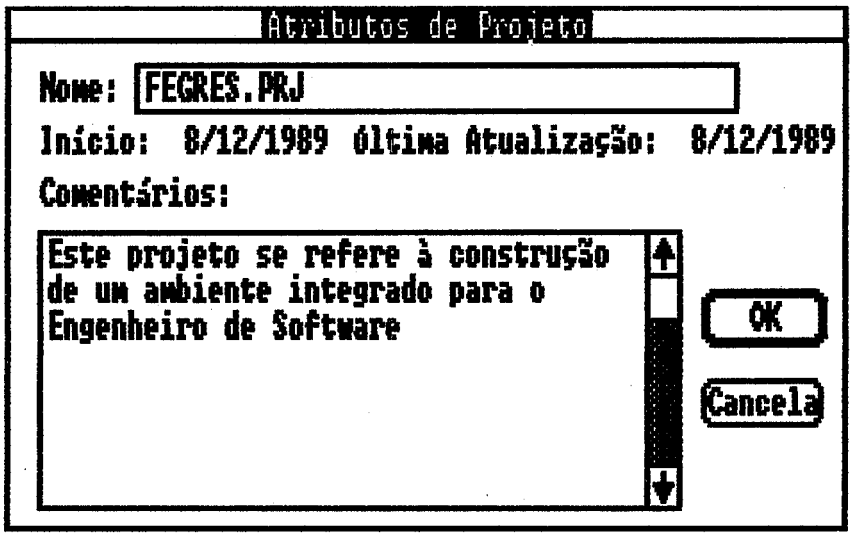

Figura A.5 - Caixa de Diálogo

5) O Mouse

Apesar de não fazer parte da interface visual o mouse é o principal dispositivo de interação e o responsável pela facilidade de uso da interface. É ele que proporciona ao usuário condições satisfatórias de todos os elementos presentes na interface. É através dele que conseguimos ativar uma janela, mudar a posição de um objeto, marcar uma posição, enfim, toda a manipulação da interface é feita de maneira simplificada por ele. Não que o Windows não possa ser utilizado sem mouse mas é que toda a flexibilidade da interface reside no fato de podermos apontar para um objeto e mandar que o mesmo execute determinada tarefa.

Podemos realizar com o mouse quatro operações básicas, FOLEY [45]:

1) apontar: mover o mouse para uma determinada posição;

2) clicar: apertar o botão do mouse que está localizado numa dada posição, isto marcará esta posição;

3) carregar ("dragging"): apertar o botão do mouse e movê-lo, carregando o objeto que estiver na posição em que foi apertado o botão, e;

4) duplo clique: clicar rapidamente duas vezes o botão do mouse que está numa dada posição.

Encontramos vários modelos de mouse que podem ser utilizados. Existem aqueles que possuem três botões, outros que possuem dois e aqueles que possuem apenas um. A maioria das aplicações necessitam de apenas um botão para que posam ser utilizadas satisfatoriamente. Aplicações para o Windows normalmente se utilizam de no máximo dois botões para interagir com o usuário.

A.5.2) Atributos Principais do WINDOWS 2.0

A.5.2.1)A Máquina Virtual WINDOWS

A máquina virtual WINDOWS extende a funcionalidade do MS-DOS, QUEDENS[69]. Ao desenvolvermos uma aplicação para o WINDOWS não precisamos nos preocupar em desenvolver dispositivos de controle para os vários recursos disponíveis no ambiente.Estes controles já estão disponíveis no ambiente através de módulos apropriados fornecidos pelo fabricante do dispositivo. Isto faz com que nossas aplicações se tornem independentes dos dispositivos existentes no hardware. Assim se quisermos utilizar uma determinada impressora ou um modelo de ploter diferente basta que exista o driver correspondente do dispositivo disponível no ambiente.

Três módulos principais, KERNEL, USER e GDI realizam o trabalho de gerenciamento dos dispositivos. QUEDENS [69]

O módulo KERNEL controla e aloca todos os recursos disponíveis na máquina. Ele trabalha junto com o DOS para carregar aplicações, gerenciar memória e executar o

escalonamento das tarefas.

O módulo USER é responsável pela criação e manutenção (movimento, tamanho, ou destruição) das janelas no dispositivo de apresentação (vídeo). Realiza também o controle de ícones, cursores e envia mensagens do mouse, teclado e registradores de tempo para a aplicação.

O módulo GDI é uma linguagem gráfica com inúmeros recursos, que executa, ou simula (através da utilização de uma série de operações simplificadas), as operações gráficas necessárias para criar imagens em todos os dispositivos de apresentação.

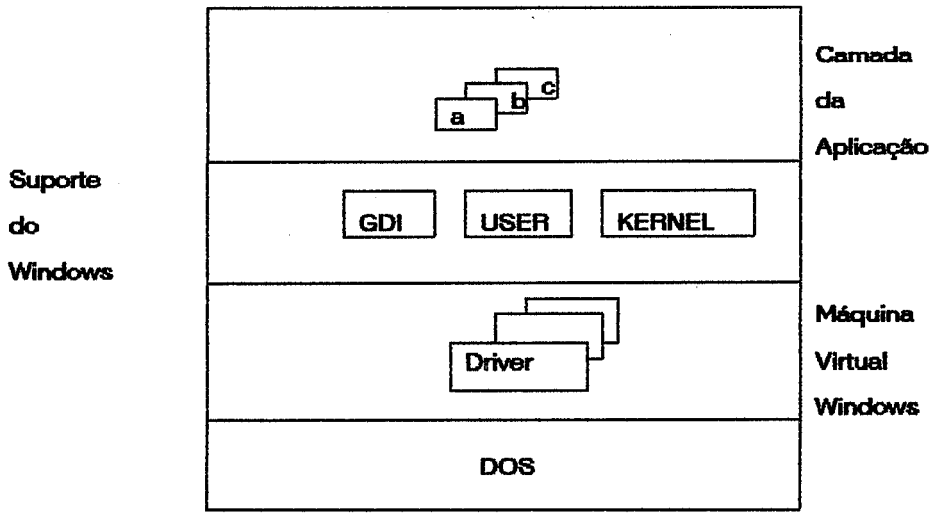

Figura A.6 - A Máquina Virtual Windows

Tanto o WINDOWS, como as aplicações desenvolvidas para ele, não se comunicam diretamente com os dispositivos de hardware. Ao invés disso elas trabalham juntas como uma máquina virtual, acessando os vários drivers supridos pelos fabricantes que compõem esta máquina . Devido aos drivers serem fornecidos pelos fabricantes dos dispositivos qualquer alteração que seja realizada neles é prontamente incorporada ao ambiente através da

substiutição do driver correspondente, evitando que toda a aplicação seja modificada e reduzindo-se o custo de manutenção. A forma como é vista pela aplicação esta máquina virtual pode ser encontrada na figura (A.6).

A.5.2.2) Sistema de Troca de Mensagens

O WINDOWS é um ambiente que simula e explora características de ambientes multi-tarefa. É possível termos mais de uma aplicação em execução dividindo os recursos do computador, MYERS [70], PETZOLD [59]. Esta característica multi tarefa é obtida através do sistema de troca de mensagens.

O sistema de troca de mensagens é composto de uma fila, onde as mensagens, inclusive as do próprio WINDOWS, são acumuladas e distribuídas conforme a aplicação a que se destinam. Uma mensagem é considerada como qualquer evento que é importante para o sistema , como por exemplo pressionar uma tecla ou movimentar o mouse. TOWSEND[67]

Uma aplicação especialmente construída para o Windows deve de tempos em tempos verificar sua fila de mensagens para averiguar a existência de alguma mensagem a espera de ser executada. O ato da aplicação consultar sua fila de mensagens fornece ao Windows a possibilidade de assumir o controle por alguns instantes e buscar a próxima mensagem da fila do sistema e enviá-la para a a fila da aplicação correspondente. Desta forma nenhuma aplicação detêm a cpu exclusivamente, a não ser que seja uma aplicação mal projetada, e aí podem ocorrer várias aplicações compartilhando o sistema. Aplicações convencionais não realizam esta tarefa e por isso não conseguem compartilhar e utilizar os recursos disponíveis no ambiente.

As mensagens podem vir de diversas fontes. O maior gerador de mensagens é o próprio Windows. Devido ao fato dele ter que controlar toda a aparência das janelas e ao mesmo tempo suportar as constantes modificações de apresentação e necessidades das várias

aplicações ele fica disparando mensagens para todas as janelas e aplicações que estiverem carregadas na memória naquele instante.

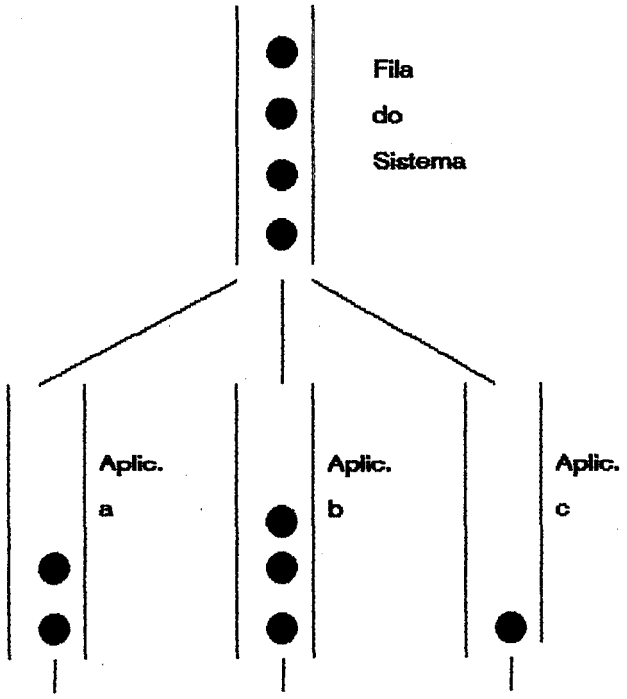

Figura A.7 - Filas de Mensagens no Windows

A.5.2.3) Gerenciamento de Memória

Devido a necessidade de compartilhamento da memória entre várias aplicações, o WINDOWS possui um gerenciamento de memória próprio, compatível com o sistema operacional OS/2, QUEDENS[69], SCHILDT [71], FOX[72]. A memória é particionada em blocos contínuos de tamanho variável, denominados segmentos, MICROSOFT [68]. Estes segmentos podem conter código executável e dados para a aplicação.

Podemos classificar os segmentos de acordo com as carcterísticas que o mesmo pode

assumir numa aplicação:

.FIXED: o segmento fica fixo em uma certa posição de memória. Devido ao Windows necessitar a todo instante deslocar blocos de memória no sentido de otimizar o uso de memória , a utilização deste tipo de segmento é desaconselhada;

.MOVEABLE: o segmento pode ser livremente deslocado na memória, permitindo uma melhor distribuição da mesma;

.LOADONCALL: o segmento fica residindo em memória auxiliar e só é carregado quando necessário;

.PRELOAD: o segmento é imediatamente carregado quando da execução da aplicação;

.SHARED/SINGLE: o segmento é compartilhado entre as várias instâncias da mesma aplicação;

.NOSHARED/MULTIPLE: o segmento não é compartilhado entre as várias instâncias da mesma aplicação;

.DISCARDABLE: o segmento é descartado, e carregado novamente quando necessário, se o WINDOWS precisar alocar uma determinada quantidade de memória não disponível no momento.

Para que ocorra um efetivo controle da memória o Windows necessita a todo instante realizar movimentações de segmentos para poder carregar e alocar memória para aqueles segmentos que não estão em memória no momento, apesar da aplicação a que eles pertençam estar executando. A carga de um segmento do disco para a memória é realizada da seguinte

forma:

1. é verificada a existência de algum segmento de memória livre que possa caber o segmento a ser carregado (técnica First Fit);

2. caso exista área carrega o segmento do disco e continua os trabalhos;

3. se não existe área livre, move todos os segmentos marcados como MOVEABLE, tentando compactar os segmentos livres de maneira a obter um bloco contíguo de memória livre, volta ao passo 1;

4. se ainda assim não conseguiu área livre, descarta todos os segmentos marcados como DISCARDABLE utilizando a técnica LRU, volta ao passo 1;

5. não conseguindo alocar o segmento, apresenta uma mensagem de memória insuficiente.

O gerenciamento de memória do WINDOWS é extremamente eficiente. Não encontramos no ambiente Windows recursos do tipo áreas de "overlay" para conseguirmos executar aplicações maiores que o montante de memória disponível. Através de um bom dimensionamento na construção da aplicação, definindo-se com clareza quais segmentos e que tipos vão assumir,conseguimos executar aplicações de tamanho muitas vezes superior a memória existente. Apesar de toda esta facilidade por parte do Windows, a performance de execução é seriamente ameaçada quando o Windows necessita fazer muitos acesso para alocar espaço.

A.5.2.4) Bibliotecas Dinâmicas

Bibliotecas dinâmicas, comumentemente conhecidas como DLL's ("Dynamic Link Libraries"), são visualizadas como entidades separadas, que não podem executar por si só, e que possuem características próprias que devem ser observadas na sua construção. MICROSOFT [88], DRAGANZA [73], JONHSON [74].

A utilização de DLL's é amplamente utilizada pelo ambiente. Os módulos KERNEL, USER e GDI , elementos principais do ambiente, são construídos como DLL's. Sua principal vantagem é evitar que unidades comuns de código sejam repetidas entre várias instâncias da mesma aplicação. Através delas podemos compartilhar código, dados e trocar informações entre as várias aplicações que estão executando no momento. A relação de funções existentes numa biblioteca é dado a partir de arquivos de definição utilizados pelo linkeditor para construir a interface da biblioteca com as aplicações. A ligação das funções é realizada em tempo de execução e não em tempo de compilação.

As funções numa DLL são reconhecidas através de identificadores numéricos especialmente nomeados na compilação da biblioteca e que possibilitam uma rápida referência durante a execução. Na realidade se comportam como ponteiros para a região dentro do segmento de código da biblioteca onde começa o código executável referente aquela função.

Os segmentos de dados e código de uma biblioteca também devem ser identificados conforme citado anteriormente para proporcionar ao Windows uma forma adequada de reconhecê-los e movê-los quando da necessidade de alocação de memória.

A.5.2.5) O Arquivo de Recursos

Uma aplicação construída para o WINDOWS se utiliza de grande parte dos recursos de interface oferecidos pelo ambiente. É comum, e aconselhável, que uma aplicação possua e utilize ícones, cursores, cardárpios e caixas de diálogo na apresentação de sua interface com o usuário. MICROSOFT [25]

Para que se consiga um melhor aproveitamento dos recursos de interface oferecidos e se obtenha uma maneira mais simples de construção e representação destes recursos por parte do desenvolvedor da aplicação, o WINDOWS se utiliza de um arquivo denominado arquivo de recursos ("resource file"), MICROSOFT [68], PETZOLD[59], para sua descrição e armazenamento.

Este arquivo pode ser construído através de ferramentas existentes no ambiente ou a partir de um editor de texto comum, desde que se observe a sintaxe apropriada para sua construção. É possível que neste arquivo sejam descritos cursores, ícones, fontes, cardápios, aixas de diálogo, mapas de bits ("bitmaps"), aceleradores, tabelas de caracteres e tabelas de dados para a aplicação. Possibilita ainda que o desenvolvedor da aplicação construa seu próprio recurso de interface, aumentando assim, as formas de armazenamento possíveis e incrementando as características do ambiente.

Este arquivo de recursos é compilado através de um compilador próprio e seu código é agregado a aplicação. O fato de agregarmos este código a aplicação fará com que todas as outras instâncias desta aplicação que por ventura sejam carregadas para execução compartilhem deste código, não necessitando duplicá-lo para poder utilizá-lo.

A.5.2) A Interface Comum Existente nas Ferramentas

A interação do usuário com as ferramentas componentes do FEGRES é feita através do mouse. Apesar de ser possível utilizar as ferramentas sem este dispositivo esta prática não é muito aconselhável pois provoca uma perda de flexibilidade no uso dos recursos do ambiente.

As janelas principais que pertencem às aplicações possuem uma barra de título contendo o nome da aplicação juntamente com o nome do projeto que se está trabalhando no momento. Abaixo desta barra de título uma barra de cardápio se apresenta com as várias opções de trabalho possíveis de serem realizadas. Estas opções podem ser ativadas apertando-se a tecla <ALT> juntamente com a letra correspondente que se encontra grifada, ou posicionando-se o cursor sobre a opção e apertando o botão esquerdo do mouse. Caso se deseje cancelar uma dada

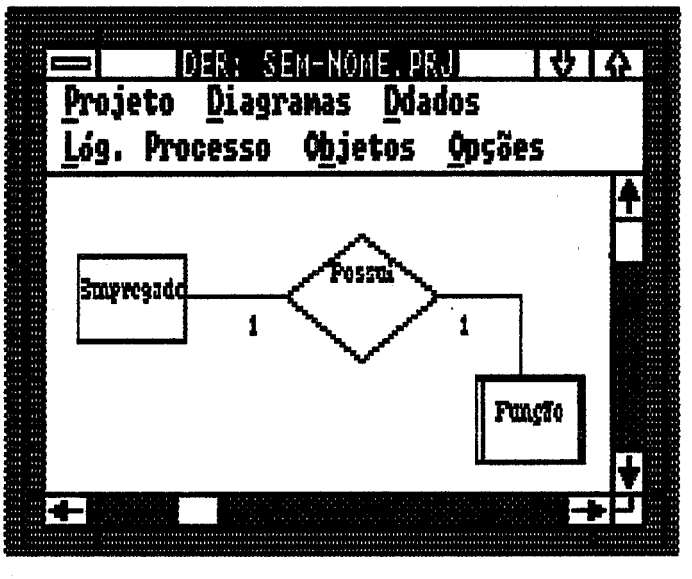

Figura A.8 - Uma Janela de Aplicação de FEGRES

opção deve-se teclar <ESC> ou então levar o cursor para uma região fora da opção e apertar novamente o botão esquerdo do mouse. Ao se escolher uma opção da barra de cardápios um cardápio secundário se apresentará ao usuário para a escolha da atividade que deseja realizar. Nestes cardápios secundários encontramos nomes de opções escritas em preto e em cinza. As opções escritas em preto podem ser usadas e estão disponíveis na ferramenta. Aquelas opções

escritas em cinza estão desabilitadas naquele instante, ou ainda não estão implementadas, e sua escolha não provocará qualquer ação. Na figura A.8 podemos ver a janela principal de uma aplicação componente do FEGRES.

As opções existentes no cardápio principal foram separadas observando-se a que objeto aquela opção estaria diretamente ligada. Existem seis opções principais as quais são: Projeto, Diagramas, DDados, Lógica dos Processos, Objetos e Opções.

A.5.2.1) A Opção Projeto

Esta opção é a porta de acesso para o objeto projeto. Através dela podemos realizar as seguintes tarefas:

1) Abrir Projeto: permite ao usuário escolher um determinado projeto contido na unidade que estiver ativa naquele instante. É possível ao usuário percorrer todas as unidades e pastas existentes no equipamento para efetuar a escolha de seu projeto. Ativar esta opção apresentará para o usuário a caixa de diálogo apresentada na figura A.9. Para efetuar a escolha o usuário deve posicionar o cursor sobre o nome que deseja acessar e apertar duas vezes o botão esquerdo do mouse ("double click"). A escolha do projeto implica em modificar o projeto corrente das ferramentas que estejam conectadas no momento.

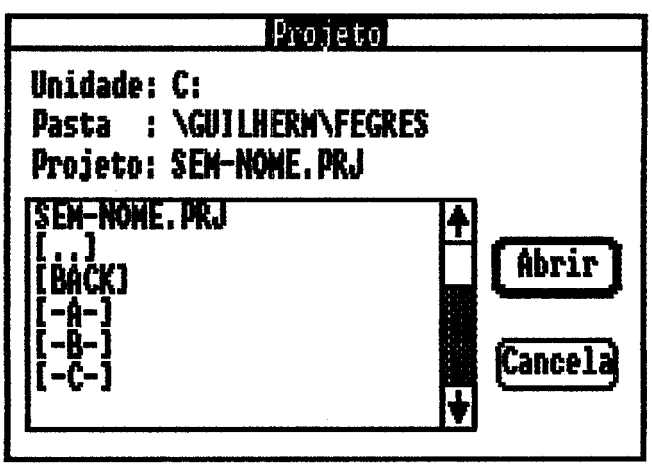

Figura A.9 - Caixa de Diálogo Abrir Projeto

2) Atributos: permite ao usuário modificar alguns atributos do projeto. Atributos possíveis de serem modificados são o nome do projeto e observações a respeito do projeto. Um nome de projeto pode conter no máximo 20 caracteres. Observações a respeito do projeto são editadas numa caixa de edição apropriada e podem conter no máximo uma página de texto, equivalente a 2 000 caracteres. Na figura A.10 podemos ver a caixa de diálogo para mudança dos atributos do projeto.

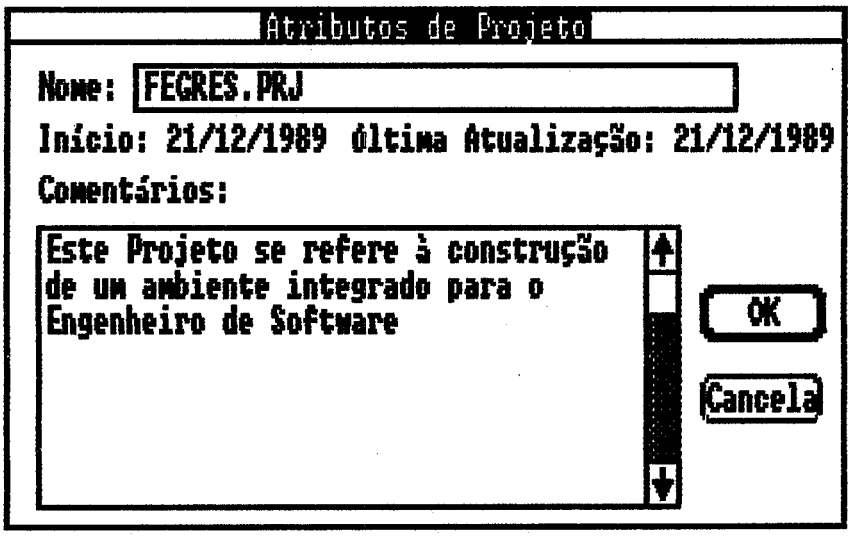

Figura A.10 - Caixa de Diálogo para Atributos de Projeto

3) Salvar: permite ao usuário guardar o projeto na pasta correspondente. Esta opção será automaticamente ativada quando o usuário tentar encerrar a utilização da ferramenta e o projeto associado tiver sido de alguma forma modificado.

4) Salvar como: permite ao usuário guardar o projeto correspondente com outro nome. Esta função é automaticamente ativada quando o usuário requisita a ferramenta que guarde um projeto sem nome. Na figura A.11 podemos verificar a caixa de diálogo para mudar o nome do projeto.

5) Imprimir: permite ao usuário imprimir as informações manipuladas pela ferramenta. Esta impressão é feita no dispositivo que estiver associado como impressora no equipamento. Na

figura (A.12) podemos ver a caixa de diálogo da impressão.

Figura A.11 - Caixa de Diálogo Salvar Como

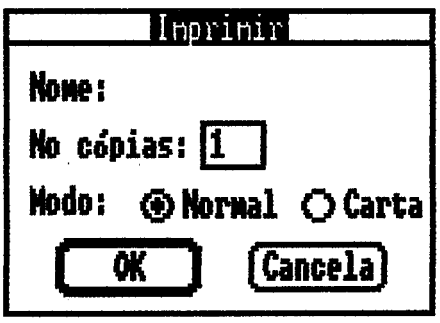

Figura A.12 - Caixa de Diálogo para Impressão

6) Reconstituir: permite ao usuário restaurar as informações relativas ao projeto até a última atualização realizada. Toda alteração realizada até então que não tenha sido preservada será perdida.

7) Copiar p/ Rascunho: permite ao usuário fazer uma cópia do projeto que está trabalhando para uma área de rascunho. Em implementação.

8) Trocar com Rascunho: permite ao usuário trocar a versão corrente do projeto por aquela que foi anteriormente copiada para o rascunho. Em implementação.

9) Esvaziar: permite ao usuário limpar todas as informações relativas ao projeto que estiver

editando. Está função provoca a criação de um projeto vazio. Em implementação.

A.5.2.2) A Opção Diagramas

O cardápio desta opção é composto de dois ícones identificando os possíveis diagramas editáveis por FEGRES. A escolha de um determinado ícone fará com que seja ativado o editor de diagramas correspondentes.

A.5.2.3) A Opção DDados

Esta opção permite ao usuário acessar no dicionário de dados as fichas correspondentes aos objetos manipulados pela ferramenta. As opções que pertencem aos objetos manipulados pela ferramenta são escritos em preto enquanto aquelas opções que não pertencem são escritas em cinza. Ao se escolher uma destas opções o dicionário de dados apresentará a ficha correspondente ao elemento que esteja selecionado neste instante.

A.5.2.4) A Opção Lógica Processos

Esta opção permite ao usuário acessar as ferramentas que descrevem a lógica detalhada de um processo. Suas opções estão escritas em cinza devido às ferramentas não estarem implementadas.

A.5.2.5) A Opção Objetos

A escolha desta opção apresentará ao usuário um cardápio com uma série de ícones representando os vários objetos que a ferramenta pode tratar. Apenas os editores gráficos possuem esta opção. A escolha de um objeto deste cardápio fará com que um objeto deste tipo seja incluído na última posição marcada. Na figura A.13 podemos ver os ícones de todos os

objetos que podem ser representados por FEGRES.

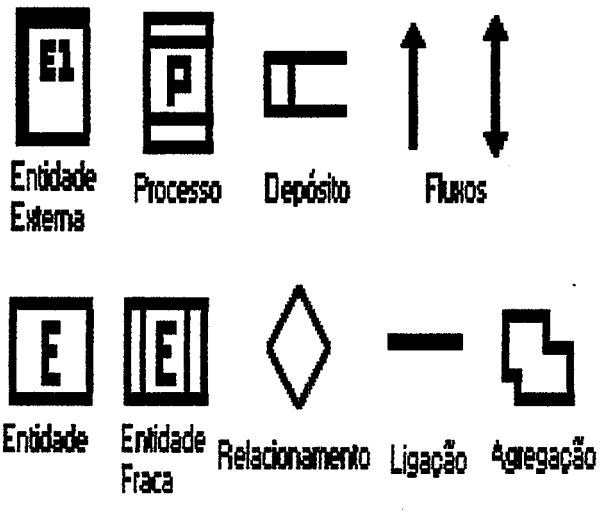

Figura A.13 - Objetos Representados por FEGRES

A.5.2.6) A Opção Opções

Está opção permite ao usuário ajustar uma série de características de trabalho nas ferramentas. Através dela pode-se realizar as seguintes tarefas:

1) Grid: permite ao usuário marcar toda a região de traçado com pontos equidistantes que facilitam o enquadramento do desenho. A figura A.14 mostra uma ferramenta com a opção Grid definida. A escolha desta opção por parte do usuário fará com que uma marca seja colocada na frente da opção representando que a mesma está ativa.

2) Zoom: permite ao usuário visualizar um pedaço da folha de traçado. A ativação do zoom colocará disponível para o usuário barras de rolamento na dimensão horizontal e vertical que permitirão percorrer todo o desenho. Como na opção de Grid uma marca será colocada junto à opção para representar que a mesma está ativa. A figura A.15 mostra um editor com a opção de zoom ativa.

3) Retirar Elemento: permite ao usuário retirar um determinado elemento do desenho. Para que isto ocorra é necessário que o elemento tenha sido anteriormente selecionado. A utilização desta opção fará com que toda a informação relativa a este elemento seja retirada da base de dados.

4) Desfazer: permite ao usuário cancelar a última operação que tenha sido realizada. No momento está implementada apenas para desmarcar pontos.

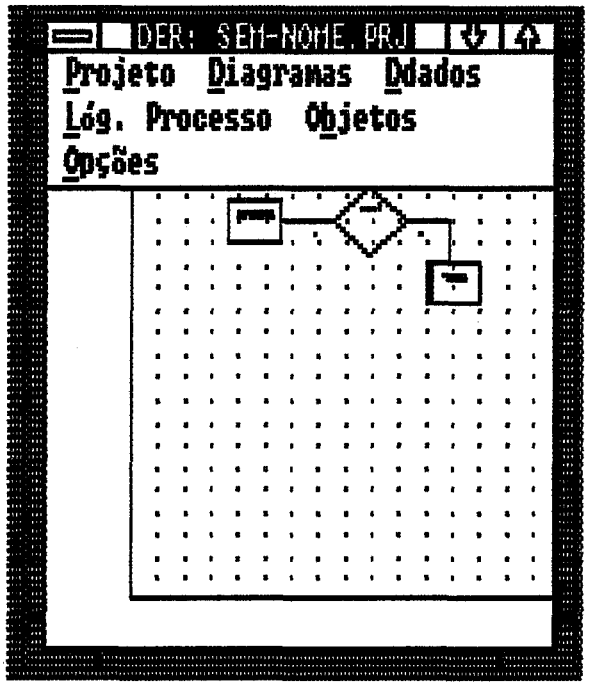

Figura A.14 - Ferramenta com opção Grid

5) Redesenhar: permite ao usuário redesenhar o diagrama que está editando.

6) 80/132 Colunas: permite ao usuário marcar as dimensões do formulário em que ele irá imprimir os diagramas. Estas dimensões são representadas como um retângulo proporcional ao tamanho de uma folha de formulário. A região externa a este retângulo pode ser utilizada para

desenho mas não será impressa.

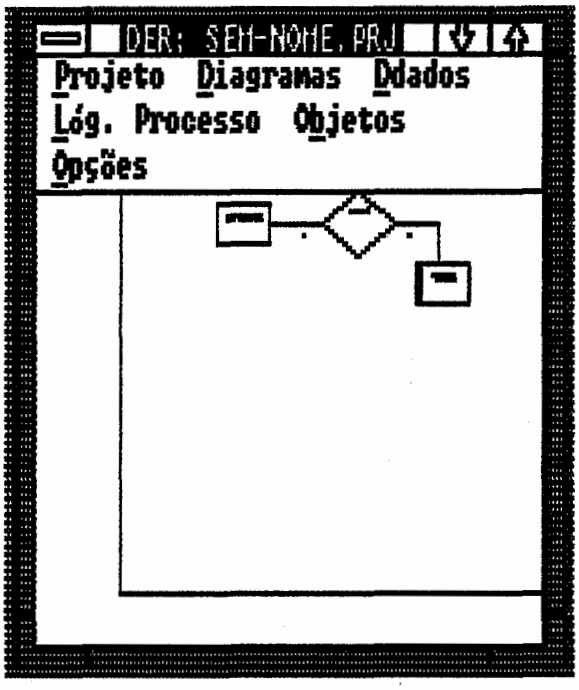

Figura A.15 - Ferramenta com opção Zoom

7) Traçado: permite ao usuário ativar e desativar uma caixa de diálogo para escolha do tipo de traçado que pode assumir um fluxo de dados ou uma ligação. A ativação desta opção provocará o aparecimento de uma marca ao lado da opção e ao mesmo tempo o aparecimento de uma janela contendo as várias opções de traçado. A figura A.16 mostra a aparência desta janela. Para se fazer uma seleção do tipo de traçado basta posicionar o cursor sobre o traçado e clicar uma vez. Fazendo isto, a região que contém o traçado escolhido será apresentada em modo reverso. Este tipo de traçado permanecerá até que uma nova escolha seja feita.

Figura A.16 - Janela para escolha do Percurso

A.5.3) O Tratamento dos Objetos Gráficos nos Editores de Diagramas de FEGRES

A.5.3.1) Incluir Elementos

1) Processo: a inclusão de um Processo é feita marcando-se a posição onde desejamos incluir o Processo e escolhendo este objeto na barra de cardápios Objetos do editor de DFD. Esta posição é marcada posicionando-se o cursor sobre o ponto desejado e clicando o botão esquerdo do mouse. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para descrição do elemento pela ferramenta. São consideradas informações mínimas para representação do Processo, seu Nome e Identificação.

2) Entidade Externa: a inclusão de uma Entidade Externa é feita marcando-se a posição onde desejamos incluir a Entidade Externa e escolhendo este objeto na barra de cardápios Objetos do editor de DFD. Esta posição é marcada posicionando-se o cursor sobre o ponto desejado e clicando o botão esquerdo do mouse. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação da Entidade Externa, seu Nome e Identificação. Caso já exista alguma Entidade Externa com esta identificação este novo elemento será considerado como uma repetição do anterior e será representado com as marcas de repetição correspondentes.

3) Depósito de Dados: a inclusão de um Depósito de Dados é feita marcando-se a posição onde desejamos incluir o Depósito de Dados e escolhendo este objeto na barra de cardápios Objetos do editor de DFD. Esta posição é marcada posicionando-se o cursor sobre o ponto desejado e clicando o botão esquerdo do mouse. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação do

Depósito de Dados, seu Nome e Identificação. Caso já exista algum Depósito de Dados com esta identificação este novo elemento será considerado como uma repetição do anterior e será representado com as marcas de repetição correspondentes.

4) Fluxo de Dados: a inclusão de um Fluxo de Dados é feita marcando-se a origem e o destino do Fluxo juntamente com o tipo de percurso que deve ser associado ao mesmo. A marcação da origem ou do destino é realizada posicionando-se o cursor o mais próximo possível do objeto que se deseja ser a origem ou o destino e clicando-se o botão direito do mouse. A ordem dos pontos é importante, neste caso, pois o primeiro ponto sempre será considerado a origem. O tipo de percurso é obtido a partir da escolha da representação correspondente na opção Percursos. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação do Fluxo de Dados seu Nome. A descrição da origem e do destino é fornecida automaticamente pela ferramenta;

5) Entidade: a inclusão de uma Entidade é feita marcando-se a posição onde desejamos incluir a Entidade e escolhendo este objeto na barra de cardápios Objetos do editor de DER. Esta posição é marcada posicionando-se o cursor sobre o ponto desejado e clicando o botão esquerdo do mouse. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação da Entidade, o seu Nome.

6) Relacionamento: a inclusão de um Relacionamento é feita marcando-se a posição ondee desejamos incluir o Relacionamento e escolhendo este objeto na barra de cardápios Objetos do editor de DER. Esta posição é marcada posicionando-se o cursor sobre o ponto desejado e clicando o botão esquerdo do mouse. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação do Relacionamento, o seu Nome.

7) Ligação: a inclusão de uma Ligação é feita marcando-se a origem e o destino da Ligação juntamente com o tipo de percurso que deve ser associado a ela. A marcação da origem ou do destino é realizada posicionando-se o cursor o mais próximo possível do objeto que se deseja ser a origem ou o destino e clicando o botão direito do mouse. A ordem dos pontos é importante, neste caso,pois o primeiro ponto sempre será considerado a origem. O tipo de percurso é obtido a partir da escolha da representação correspondente na opção Percursos. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação da Ligação sua cardinalidade e totalidade. A descrição da origem e do destino é fornecida automaticamente pela ferramenta.

8) Agregação: a inclusão de uma Agregação é feita marcando-se os dois pontos que definem a região retangular da agregação.Estes pontos são marcados posicionando-se o cursor na posição desejada e apertando o botão direito do mouse. A ordem dos pontos não é importante neste caso. Automaticamente uma ficha de descrição para o elemento será aberta pelo Dicionário de Dados para que o usuário forneça as informações mínimas para sua apresentação. São consideradas informações mínimas para representação da Agregação, seu Nome.

A.5.3.2) Alterar Elementos

Dois tipos de alteração podem ser realizadas nos objetos: alteração da posição e alteração dos dados de descrição. A alteração dos dados de descrição é realizada a partir do Dicionário de Dados e será tratada posteriormente.

1) Processo: para alterar a posição de um Processo devemos posicionar o cursor sobre o Processo desejado e apertar o botão esquerdo do mouse mantendo-o neste estado. Uma região tracejada correspondente à área do Processo será apresentada. Movemos esta área tracejada para a posição desejada e soltamos o botão do mouse que estava apertado. O Processo

será colocado nesta nova posição. Os Fluxos de Dados associados com ele também serão transferidos;

2) Entidade Externa: para alterar a posição de uma Entidade Externa devemos posicionar o cursor sobre a Entidade Externa desejada e apertar o botão esquerdo do mouse mantendo-o neste estado. Uma região tracejada correspondente à área da Entidade Externa será apresentada. Movemos esta área tracejada para a posição desejada e soltamos o botão do mouse que estava apertado. A Entidade Externa será colocada nesta nova posição. Os Fluxos de Dados associados com ela também serão transferidos;

3) Depósito de Dados: para alterar a posição de um Depósito de Dados devemos posicionar o cursor sobre o Depósito de Dados desejado e apertar o botão esquerdo do mouse mantendo-o neste estado. Uma região tracejada correspondente à área do Depósito de Dados será apresentada. Movemos esta área tracejada para a posição desejada e soltamos o botão do mouse que estava apertado. O Depósito de Dados será colocado nesta nova posição. Os Fluxos de Dados associados com ele também serão transferidos;

4) Fluxo de Dados: os Fluxos de Dados não podem ser alterados diretamente devido a estarem ligados aos seus objetos origem e destino. Entretanto, para facilitar um ajuste visual do desenho, é possível alterar a posição da descrição do Fluxo de Dados. Para isto devemos posicionar o cursor sobre o texto que descreve o Fluxo de Dados e apertar o botão esquerdo do mouse. Uma região tracejada correspondente à área do texto será apresentada. Movemos esta região para a posiçáso desejada e soltamos o botão do mouse. O texto será posicionado nesta nova posição;

5) Entidade: para alterar a posição de uma Entidade devemos posicionar o cursor sobre a Entidade desejada e apertar o botão esquerdo do mouse mantendo-o neste estado. Uma região tracejada correspondente à área da Entidade será apresentada. Movemos esta área

tracejada para a posição desejada e soltamos o botão do mouse que estava apertado. A Entidade será colocada nesta nova posição. As Ligações associadas a ela também serão transferidas;

6) Relacionamento: para alterar a posição de um Relacionamento devemos posicionar o cursor sobre o Relacionamento desejado e apertar o botão esquerdo do mouse mantendo-o neste estado. Uma região tracejada correspondente à área do Relacionamento será apresentada. Movemos esta área tracejada para a posição desejada e soltamos o botão do mouse que estava apertado. O Relacionamento será colocado nesta nova posição. As Ligações associadas a ele também serão transferidas;

7) Ligação: as Ligações não podem ser alteradas diretamente devido a estarem ligadas aos seus objetos origem e destino. Entretanto, para facilitar um ajuste visual do desenho, é possível alterar a posição da cardinalidade da Ligação. Para isto devemos posicionar o cursor sobre o texto que descreve a cardinalidade da Ligação e apertar o botão esquerdo do mouse. Uma região tracejada correspondente à área do texto será apresentada. Movemos esta região para a posição desejada e soltamos o botão do mouse. O texto será posicionado nesta nova posição;

8) Agregação: para alterar a posição de uma Agregação devemos posicionar o cursor sobre a área livre pertencente à Agregação e apertar o botão esquerdo do mouse. Uma região tracejada correspondente à área total da Agregação será apresentada. Movemos esta região para a posição desejada e soltamos o botão do mouse. A Agregação será posicionada na nova posição juntamente com todos os elementos que a compõem.

A.5.3.3) Excluir Elementos

A exclusão dos objetos de FEGRES é feita selecionando-se o objeto e escolhendo a opção Retirar Elemento do cardápio Opções. Esta exclusão implicará na retirada de todas as informações referentes a este objeto da base de dados do sistema visando manter a consistência

entre os documentos.

A seleção dos objetos deve ser executada posicionando-se o cursor sobre o objeto desejado e pressionando-se ao mesmo tempo a tecla <shift> mais o botão esquerdo do mouse. Os objetos Fluxo de Dados e Ligação são identificados pelos textos que os descrevem, que são, respectivamente, a descrição do Fluxo de Dados e a cardinalidade da Ligação.

A.5.4) A Documentação dos objetos de FEGRES

Toda a informação gerada pelos editores gráficos é automaticamente armazenada na base de dados visando manter o mais atualizado possível os dados disponíveis sobre o desenvolvimento do projeto em questão. Este armazenamento é feito a partir de um pequeno gerenciador de banco de dados construído especialmente para suportar os objetos de FEGRES.

A forma de interação e verificação das informações existentes sobre o projeto em desenvolvimento pode ser realizada a partir do Dicionário de Dados. É através dele que podemos acessar as fichas de descrição dos objetos e modificar, de alguma forma, as informações contidas sobre este objeto.

Existe uma interação direta do Dicionário de Dados com os editores gráficos componentes do ambiente. Toda modificação sofrida pelos objetos nos editores gráficos é refletida imediatamente para o Dicionário de Dados, e também, toda modificação nos dados textuais efetuada pelo Dicionário de Dados são também refletidas imediatamente para a representação gráfica que os editores gráficos proporcionam.

As informações relativas ao objeto são tratadas pelo Dicionário de Dados através de fichas apropriadas. Cada ficha é projetada para poder representar a informação correspondente ao objeto que a mesma está associada. As informações são fornecidas através dos campos

existentes nesta ficha. A navegação entre estes campos pode ser feita através do mouse, posicionando-se o cursor sobre o campo e apertando obotão esquerdo do mouse, ou então, através das teclas <shift> e <tab>. Apertando-se <tab> o cursor navegará pelos campos de cima para baixo. Apertando-se <shift> + <tab> o cursor navegará de baixo para cima.

Todas as fichas apresentam três campos em comum que permitem informar a respeito do processo de edição e consulta das informações. O campo Ok deve ser informado toda vez que o processo de edição e consulta as informações tenha se realizado de maneira satisfatório pelo usuário e as informações contidas na ficha possam ser aproveitadas. O campo Próximo deve ser informado toda vez que o usuário desejar consultar as informações contidas na ficha que descreve o próximo elemento daquele tipo. O campo Cancela informa ao Dicionário para não considerar as informações existentes na ficha que por ventura tenham sido modificadas e fechar aquela ficha.

A.5.4.1) Ficha para descrição de Processo

A ficha através da qual podemos descrever as informações relativas a um Processo no Dicionário de Dados pode ser visualizada na figura A.17. Os seguintes campos compõem esta ficha:

1) Processo: contém a descrição funcional do Processo. Este campo aceita até 40 caracteres alfanuméricos;

2) Ref.: contém a referência do processo num diagrama de Fluxo de Dados. Este campo aceita até 11 caracteres alfanuméricos;

3) Descrição: contém uma descrição resumida do que o Processo representa no projeto. Este campo possui um pequeno editor de texto associado para sua edição e aceita até 254 caracteres alfanuméricos;

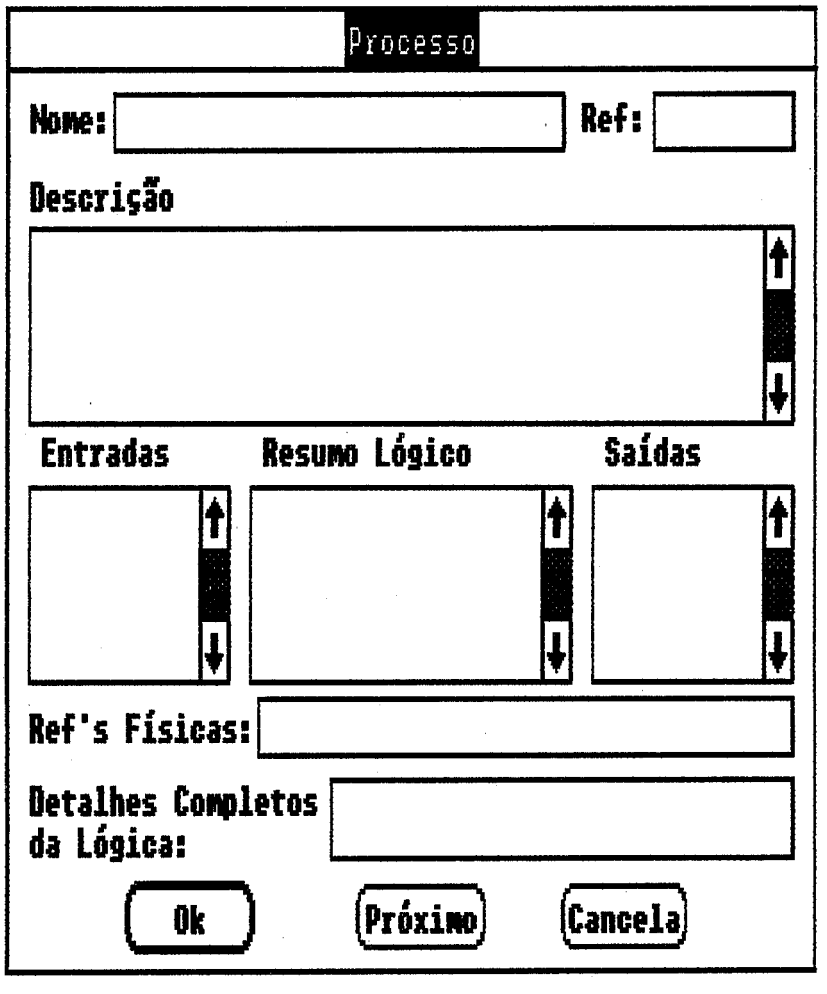

Figura A.17 - Ficha de Documentação de Processo

4) Entradas: contém referências a todas as estruturas de dados e itens de dados que entram no processo para que o mesmo possa executar suas funções. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres. Para uma perfeita representação das entradas do Processo as mesmas deverão ser separadas por ponto-e-vírgula na sua representação;

5) Resumo Lógico: contém uma pequena descrição da lógica do processo que permita identificar com mais clareza o que o processo faz. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres alfanuméricos;

6) Saídas: contém referências a todas as estruturas de dados e itens de dados que saem do processo após a execução de suas funções. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres. Para uma perfeita representação das saídas do Processo as mesmas deverão ser separadas por ponto-e-vírgula na sua representação;

7) Refs.Físicas: contém a localização física do processo. Este campo pode conter até 128 caracteres alfanuméricos;

8) Detalhes Completos da Lógica Encontram-se Em: contém referências que permitam um melhor detalhamento do processo em questão. Este campo pode conter até 128 caracteres alfanuméricos.

A.5.4.2) Ficha para Descrição de Entidade Externa

A ficha através da qual podemos descrever as informações relativas a uma Entidade Externa no Dicionário de Dados pode ser visualizada na figura A.18. Os seguintes campos compõem esta ficha:

1) Entidade Externa: contém o nome da Entidade Externa. Este campo pode conter até 40 caracteres alfanuméricos;

2) Ref.: contém a referência da Entidade Externa no projeto que está sendo desenvolvido. Este campo pode conter até 5 caracteres alfanuméricos;

3) Descrição: contém uma descrição que permita identificar de maneira clara o que a Entidade Externa representa. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres alfanuméricos;

Figura A.18 - Ficha de Descrição de Entidade Externa

4) Fluxos que Entram: contém a descrição dos fluxos de dados que chegam na Entidade Externa. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres. Para uma perfeita representação dos Fluxos que entram os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

5) Fluxos que Saem: contém a descrição dos fluxos de dados que partem da Entidade Externa. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254

caracteres. Para uma perfeita representação dos Fluxos que saem os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

6) Linguagem: contém a descrição da linguagem que foi implementada a Entidade Externa caso a mesma seja um sistema de processamento de dados. Este campo pode conter até 40 caracteres alfanuméricos;

7) Hardware: contém a descrição do equipamento onde a Entidade Externa está instalada caso seja um sistema de processamento de dados. Este campo pode conter até 40 caracteres alfanuméricos;

8) Pessoa: contém o nome do responsável pelo sistema de processamento de dados que representa a Entidade Externa. Este campo pode conter até 40 caracteres;

9) Telefone: contém o número do telefone onde pode ser encontrado o responsável pelo sistema de processamento de dados que representa a Entidade Externa. Este campo pode conter até 11 caracteres alfanuméricos;

10) Setor: contém a identifcação do setor pertencente a outra organização que represetna a Entidade Externa. Este campo pode conter até 40 caracteres;

11) Telefone: contém o número do telefone onde pode ser encontrado o responsável pela Entidade Externa em outra organizaçáo. Este campo pode conter até 11 caracteres alfanuméricos;

12) Pessoa: contém o nome do responsável pela Entidade Externa pertencente a outra organização. Este campo pode conter até 40 caracteres alfanuméricos.

A.5.4.3) Ficha para Descrição de Depósito de Dados

A ficha através da qual podemos descrever as informações relativas a um Depósito de Dados no Dicionário de Dados pode ser visualizada na figura A.19. Os seguintes campos compõem esta ficha:

1) Depósito de Dados: contém a identificação do Depósito de Dados no projeto que está sendo desenvolvido. Este campo pode conter até 40 caracteres;

2) Ref.: contém a referência do Depósito de Dados no projeto que está sendo desenvolvido. Este campo pode conter até 5 caracteres;

Figura A.19 - Ficha de Descrição de Depósito de Dados

3) Descrição: contém uma descrição que permita identificar de maneira clara o que o Depósito de Dados representa. Este campo possui um editor de textos associado para sua edição e pode conter até 254 caracteres alfanuméricos;

4) Fluxos que Entram: contém a descrição dos fluxos de dados que chegam no Depósito de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres. Para uma perfeita representação dos Fluxos que entram os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

5) Fluxos que Saem: contém a descrição dos fluxos de dados que partem do Depósito de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres. Para uma perfeita representação dos Fluxos que saem os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

6) Conteúdo: contém a descrição das estruturas de dados e elementos de dados que compõem o Depósito de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres. Para uma perfeita representação dos dados armazenados os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

7) Análise de Acesso Imediato: contém a referência para onde será encontrada uma análise de acesso imediato. Este campo pode conter até 40 caracteres;

8) Organização Física: contém a forma como deve ser organizado o Depósito de Dados. Este campo pode conter até 40 caracteres.

A.5.4.4) Ficha para Descrição de Fluxo de Dados

A ficha através da qual podemos descrever as informações relativas a um Fluxo de Dados no Dicionário de Dados pode ser visualizada na figura A.20. Os seguintes campos compõem esta

ficha:

1) Fluxo de Dados: contém o nome do Fluxo de Dados. Este campo pode conter até 40 caracteres;

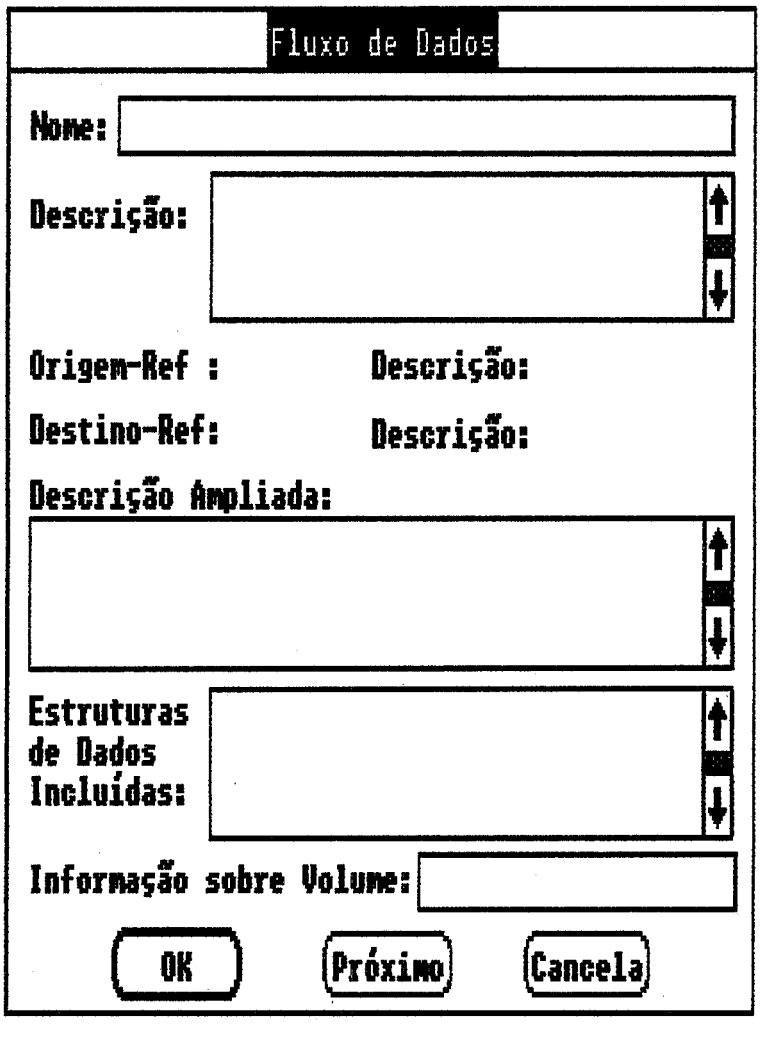

Figura A.20 - Ficha de Descrição de Fluxo de Dados

2) Descrição: contém uma descrição resumida que permita identificar o que representa o Fluxo de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 128 caracteres alfanuméricos;

3) Origem-Ref.: contém a referência para o objeto que é origem do Fluxo de Dados. Este campo pode conter até 11 caracteres;

4) Descrição: contém o nome que descreve a origem do Fluxo de Dados. Este campo pode conter até 40 caracteres;

5) Destino-Ref: contém a referência para o objeto que é destino do Fluxo de Dados. Este campo pode conter até 11 caracteres;

6) Descrição: contém o nome que descreve o destino do Fluxo de Dados. Este campo pode conter até 40 caracteres;

7) Descrição Ampliada: contém uma descrição mais detalhada que permita identificar mais precisamente o quer representa o Fluxo de Dados. Este campo possui um editor de textso associado para sua edição e pode conter até 254 caracteres alfanuméricos;

8) Estruturas de Dados Incluídas: contém a descrição das estruturas de dados e elementos de dados que são transportados pelo Fluxo de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres.Para uma perfeita representação dos dados transportados os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

9) Informação sobre Volume: contém informação a respeito do volume de informação que o Fluxo de Dados transporta. Este campo pode conter até 40 caracteres alfanuméricos.

A.5.4.5) Ficha para Descrição de Estrutura de Dados

A ficha através da qual podemos descrever as informações relativas a uma Estrutura de

Dados no Dicionário de Dados pode ser visualizada na figura A.21. Os seguintes campos compõem esta ficha:

1) Estrutura de Dados: contém o nome da Estrutura de Dados. este campo pode conter até 40 caracteres alfanuméricos;

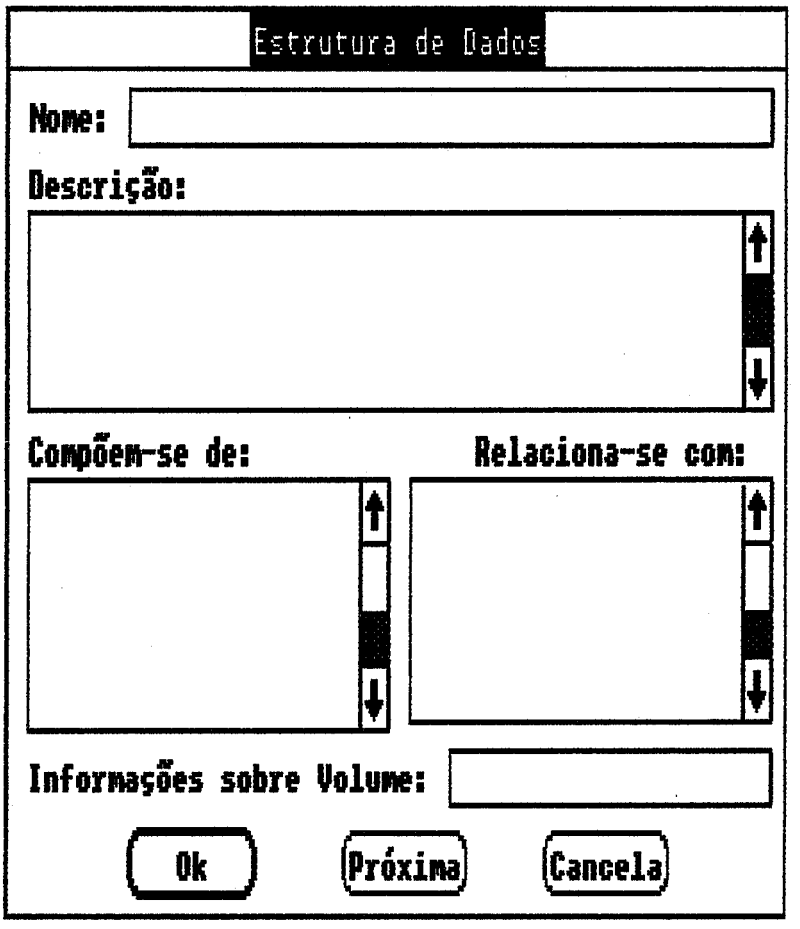

Figura A.21 - Ficha de Descrição de Estrutura de Dados

2) Descrição: contém uma descrição que permite identificar claramente o que representa a Estrutura de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres alfanuméricos;

3) Compõem-se de: contém a identificação das estruturas de dados e elementos de dados que compõem esta estrutrua. Este campo possui um pequeno editor de textos para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos elementos que compõem a Estrutura os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

4) Relaciona-se com: contém a identificação das estruturas que são compostas por esta estrutrua. Este campo possui um pequeno editor de textos para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação das Estruturas que contêm esta Estrutura as mesmas deverão ser separadas por ponto-e-vírgula na sua representação;

5) Informação sobre Volume: contém a quantidade de Estruturas de Dados deste tipo que o sistema deve trabalhar. Este campo pode conter até 40 caracteres alfanuméricos.

A.5.4.6) Ficha para Descrição de Elemento de Dados

A ficha através da qual podemos descrever as informações relativas a um Elemento de Dados no Dicionário de Dados pode ser visualizada na figura A.22. Os seguintes campos compõem esta ficha:

1) Elemento de Dados: contém o nome do Elemento de Dados. Este campo pode conter até 40 caracteres;

2) Descrição Sumária: contém uma descrição que permite identificar com clareza o Elemento de Dados. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres;

3) Tipo: contém a representação do tipo que o Elemento de Dados possui. A seleção é feita indicando qual o tipo se deseja;

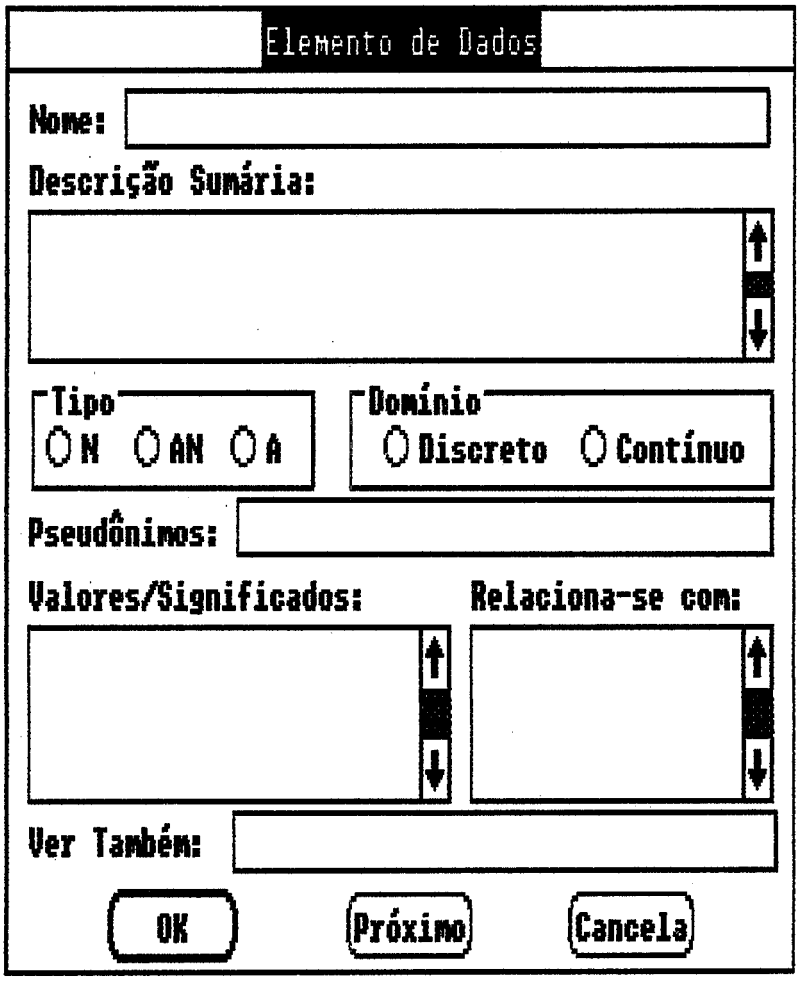

Figura A.22 - Ficha de Descrição de Elemento de Dados

4) Pseudônimos: contém os nomes que identificam o mesmo Elemento de Dados no sistema. Ester campo podê conter até 80 caracteres. Para uma melhor representação os nomes devem ser separados por ponto-e-vírgula;

5) Domínio: contém o domíniuo de valores que o Elemento de Dados pode representar. A seleção é feita indicando qual o tipo de domínio o Elemento de Dados deve representar;

6) Relaciona-se com: contém a identificação das Estruturas de Dados que são compostas pelo Elemento de Dados. Este campo possui um pequeno editor de textos para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação das Estruturas que

contêm este Elemento de Dados as mesmas deverão ser separadas por ponto-e-vírgula na sua representação;

7) Valores/Significados: contém a identificação dos possíveis valores e respectivos significados que o Elemento de Dados pode assumir caso seu domínio seja discreto. Este campo possui um pequeno editor de textos para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos pares Valor/Significado os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

8) Ver Também: contém referências onde podem ser obtidas maiores informações a respeito do Elemento de Dados. Este campo pode conter até 40 caracteres.

A.5.4.7) Ficha para Descrição de Ítem de Glossário

A ficha através da qual podemos descrever as informações relativas a um Ítem de Glossário no Dicionário de Dados pode ser visualizada na figura A.23. Os seguintes campos compõem esta ficha:

1) Ítem de Glossário: contém o nome do Ítem de Glossário. Este campo pode conter até 40 caracteres;

2) Descrição Sumária: contém uma descrição que permite identificar com clareza o Ítem de Glossário. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres;

3) Tipo: contém a representação do tipo que o Ítem de Glossário possui. A seleção é feita indicando qual o tipo se deseja;

4) Pseudônimos: contém os nomes que identificam o mesmo Ítem de Glossário no sistema. Ester

campo pode conter até 80 caracteres. Para uma melhor representação os nomes devem ser separados por ponto-e-vírgula;

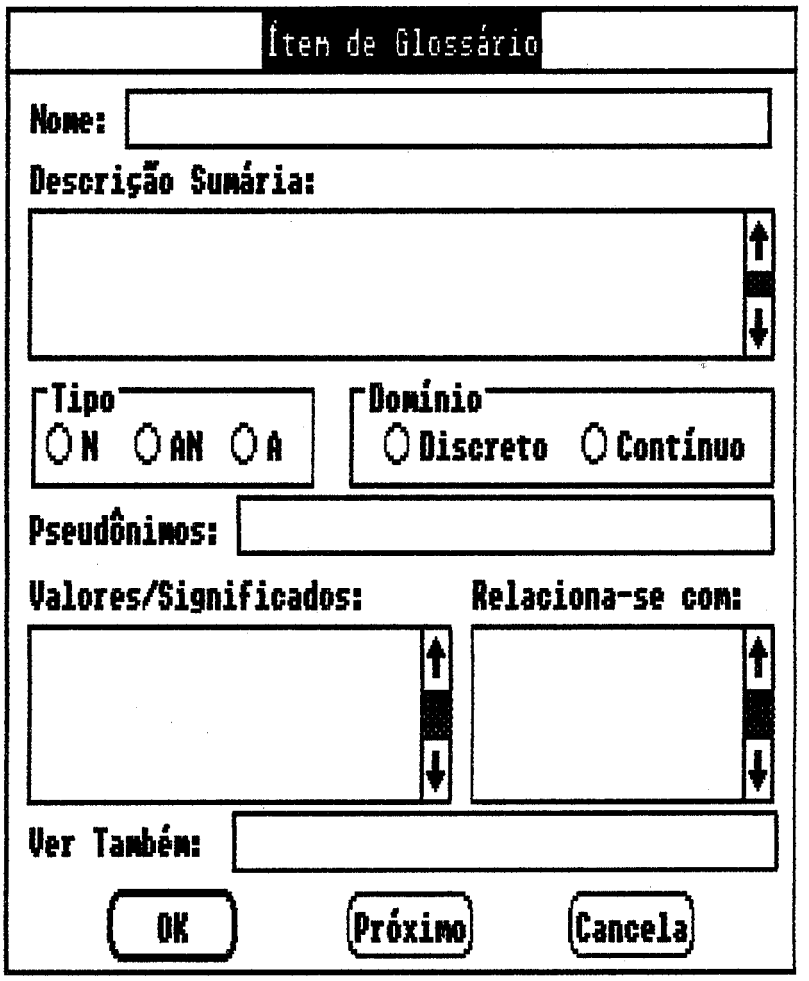

Figura A.23 - Ficha de Descrição de Ítem de Glossário

5) Domínio: contém o domíniuo de valores que o Ítem de Glossário pode representar. A seleção é feita indicando qual o tipo de domínio o Ítem de Glossário deve representar;

6) Relaciona-se com: contém a identificação das Estruturas de Dados que são compostas pelo Ítem de Glossário. Este campo possui um pequeno editor de textos para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação das Estruturas que contêm este Ítem de Glossário as mesmas deverão ser separadas por ponto-e-vírgula na sua

representação;

7) Valores/Significados: contém a identificação dos possíveis valores e respectivos significados que o Ítem de Glossário pode assumir caso seu domínio seja discreto. Este campo possui um pequeno editor de textos para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos pares Valor/Significado os mesmos deverão ser separados por ponto-e-vírgula na sua representação;

8) Ver Também: contém referências onde podem ser obtidas maiores informações a respeito do Ítem de Glossário. Este campo pode conter até 40 caracteres alfanuméricos.

A.5.4.8) Ficha para Descrição de Entidade

A ficha através da qual podemos descrever as informações relativas a uma Entidade no Dicionário de Dados pode ser visualizada na figura A.24. Os seguintes campos compõem esta ficha:

1) Nome da Entidade: contém o nome da Entidade. Este campo pode conter até 40 caracteres;

2) Determinantes: contém a identificação de todos os atributos que permitem identificar a Entidade na tabela que a representa. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos atributos os memos deverão ser separados por ponto-e-vírgula;

3) Genéricos: contém a identificação de todos os atributos que compõem a Entidade mas que não permitem realizar uma identificação da mesma na tabela que a representa. Este campo possui um pequeno editor de textos associado e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos atributos os mesmos deverão ser separados por ponto-e-vírgula.

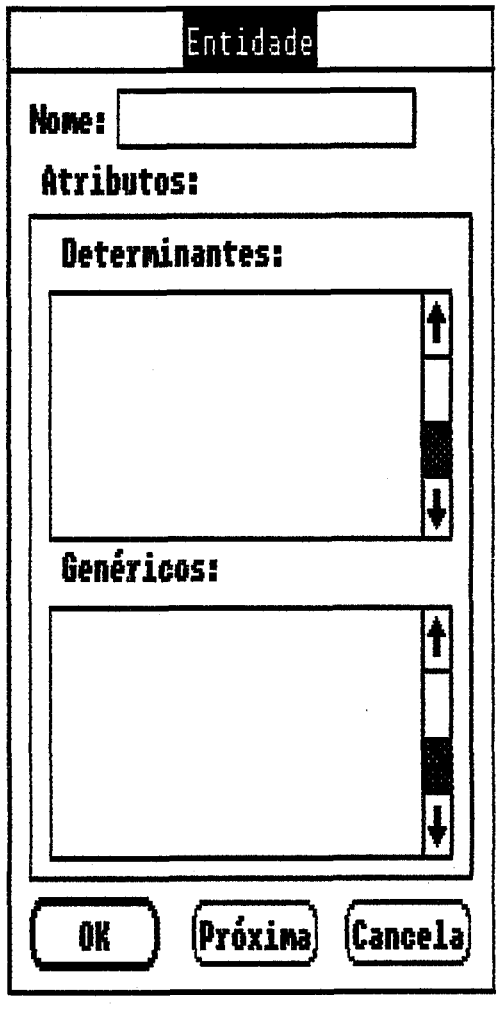

Figura A.24 - Ficha de Descrição de Entidade

A.5.4.9) Ficha para Descrição de Relacionamento

A ficha através da qual podemos descrever as informações relativas a um Relacionamento no Dicionário de Dados pode ser visualizada na figura A.25. Os seguintes campos compõem esta ficha:

1) Nome do Relacionamento: contém o nome do Relacionamento. Este campo pode conter até 40 caracteres;

2) Determinantes: contém a identificação de todos os atributos que permitem identificar o Relacionamento na tabela que o representa. Este campo possui um pequeno editor de textos associado para sua edição e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos atributos os memos deverão ser separados por ponto-e-vírgula;

3) Genéricos: contém a identificação de todos os atributos que compõem o Relacionamento mas que não permitem realizar uma identificação do mesmo na tabela que o representa. Este campo possui um pequeno editor de textos associado e pode conter até 254 caracteres alfanuméricos. Para uma perfeita representação dos atributos os mesmos deverão ser separados por ponto-e-vírgula.

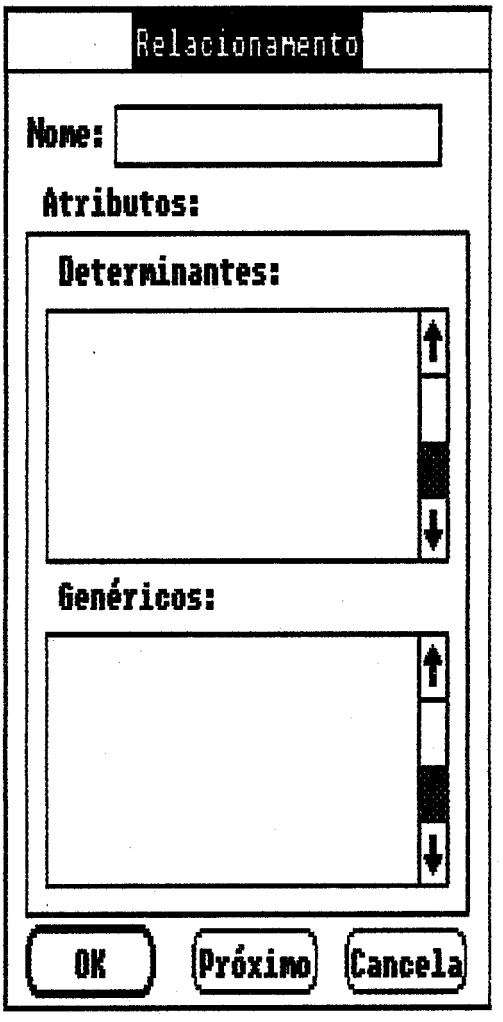

Figura A.25 - Ficha de Descrição de Relacionamento

A.5.4.10) Ficha para Descrição de Ligação

A ficha através da qual podemos descrever as informações relativas a uma Ligação no Dicionário de Dados pode ser visualizada na figura A.26. Os seguintes campos compõem esta ficha:

1) Origem: contém o nome do elemento que é origem da Ligação. Este campo pode conter até 40 caracteres alfanuméricos;

2) Destino: contém o nome do elemento que é destino da Ligação. Este campo pode conter até 40 caracteres alfanuméricos;

3) Grau: contém a identificação do grau que a ligação deve representar no relacionamento que ela conecta. A seleção deve ser feita escolhendo-se qual o grau a Ligação deve assumir;

4) Parcialidade: contém a identificação da parcialidade do relacionamento que a Ligação conecta. A seleção deve ser realizada escolhendo-se se o que a Ligação deve assumir.

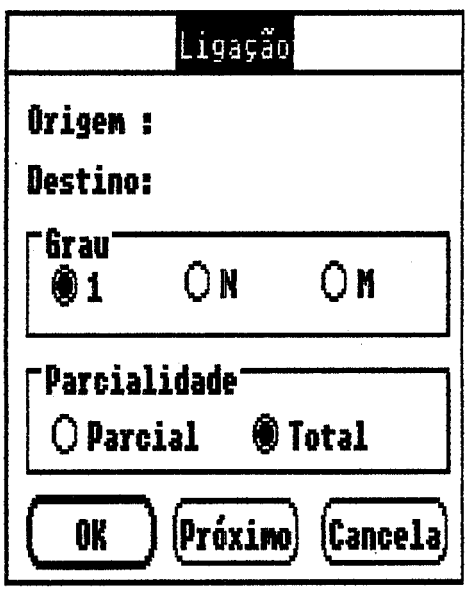

Figura A.26 - Ficha de Descrição de Ligação

A.5.4.11) Ficha para Descrição de Agregação

A ficha através da qual podemos descrever as informações relativas a uma Agregação no Dicionário de Dados pode ser visualizada na figura A.27. O seguinte campo compõem esta ficha:

1) Nome: contém o nome da Agregação. Este campo pode conter até 40 caracteres alfanuméricos.

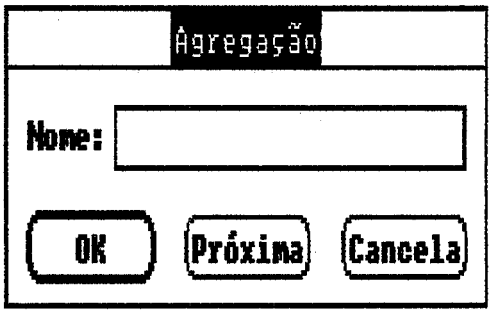

Figura A.27 - Ficha de Descrição de Agregação