# METODOLOGIA N I N D - B F B

## PROPOSTA DE UMA FERRAMENTA

YAYA TOURÉ

Orientador: Prof. Dr. Paulo Cesar Masiero

Dissertação apresentada ao ICMSC-USP
como parte de requisitos para a obtenção do título de
Mestre na área de Ciências de Computação e Matemática
Computacional.

São Carlos, Novembro de 1988

## R E S U M O

Esta dissertação apresenta a macrometodologia MIND-BFB, composta de módulos para controle de projetos, especificação e projeto de Sistemas de Informação. A metodologia para especificação, denominada REMORA é apresentada com mais detalhes e uma ferramenta de apoio ao uso dessa metodologia foi desenvolvida. A ferramenta, denominada MIND-AID, foi escrita em Prolog e apresenta facilidades para a especificação de um sistema em REMORA, análises de consistência da base de dados que representa o sistema, bem como vários relatórios para apoio na fase de especificação e nas fases do ciclo de vida.



## A B S T R A C T

re are shown the macromethodology MIND-BFB which is the tegration of three methodologies and the methodology REMORA ich is one of those three methodologies. Afterwards is given a scription of a CASE called MIND-AID, proposed to help the signer using the methodology REMORA. This CASE, composed of 11 dules and written in Prolog, permits a coherent specification the model's elements and reports generation.

## AGRADECIMENTOS:

Agradeço ao Banco Francês e Brasileiro, através da sua Diretoria de Produção, por ter me oferecido a chance excepcional de elaborar este trabalho com o seu apoio. Agradeço a todos os funcionários por terem me facilitado a estadia no banco. Agradeço especialmente a Segio Leifert por ter promovido e ajudado neste estágio, a Carlos Barbosa pela sua orientação, a Andréa Baraldi pelo trabalho inicial que me serviu de embasamento, a Jairo Avritchir e Maria Alice pelo interesse que manifestaram pelo produto o que estimulou bastante. Enfim agradeço ao Professor Paulo Masiero pela orientação acadêmica até nos fins de semana, ao senhor Cristiano Stein, ex-coordenador de convênios, e através dele a CEAG-SP, a SEPLAN e ao ITAMARATY pela bolsa de estudos concedida.

"Distingo duas maneiras de cultivar as ciências: uma é aumentar a massa de conhecimentos por decobertas e é assim que se merece o nome de "inventor"; a outra é aproximar as descobertas e ordena-las a fim de que mais homens sejam esclarecidos, e que cada um participe, segundo seu potencial, à luz do seu seculo."

DIDEROT

# I N D I C E S:


| | Página: |
|---|---|
| 1. INTRODUÇÃO | 1 |
| 1.1. Situação do trabalho | 1 |
| 1.2. Metodologias e ferramentas de apoio | 3 |
| 1.3. Organização da dissertação | 12 |
| 1.4. Conclusão | 14 |
| 2. BIBLIOGRAFIA BÁSICA E TRABALHOS CORRELATOS | 16 |
| 2.1. Introdução | 16 |
| 2.2 Publicações sobre a metodologia REMORA | 16 |
| 2.3. Publicações sobre Ferramentas de apoio | 17 |
| 2.4. Outras publicações | 21 |
| 2.5. Conclusão | 21 |
| 3. MACROMETODOLOGIA MIND-BFB | 27 |
| 3.1. Apresentação da macrometodologia | 27 |
| 3.2. Introdução à metodologia REMORA | 30 |
| 3.2.1. Problemática da concepção dos SI | 30 |
| 3.2.2. O Modelo Descritivo | 32 |
| 3.2.3. O Modelo Conceitual | 36 |
| 3.2.4. O Modelo Interno | 45 |
| 3.2.5. Um exemplo: caso de hospedagem em hotel | 51 |
| 3.3. Conclusão | 58 |
| 4. PROPOSTA DE UMA FERRAMENTA DE APOIO : MIND-AID | 59 |
| 4.1. Introdução | 59 |
| 4.2. Objetivos | 59 |
| 4.3. Arquitetura geral | 60 |
| 4.3.1. Introdução | 60 |
| 4.3.2. A Base de Dados da ferramenta | 62 |
| 4.3.3. Interface com o usuário | 63 |
| 4.3.4. Processos | 64 |
| 4.4. Descrição detalhada dos módulos | 66 |
| 4.5. Exemplo: caso de hospedagem em hotel | 74 |
| 4.6. Conclusão | 91 |
| 5. CONCLUSÃO GERAL | 92 |
| 5.1. Introdução | 92 |
| 5.2. Avaliação da Mind-Aid | 92 |
| 5.3. Desenvolvimentos Adicionais | 93 |
| BIBLIOGRAFIA | 95 |
| ANEXO 1: Glossario de abreviações | 99 |
| ANEXO 2: Sintaxe da linguagem MIND-BFB | 101 |
| ANEXO 3: Listagem comentada do módulo MC1 | 103 |
| ANEXO 4: Exemplo de Base de Dados de especificação: caso de hospedagem em hotel | 112 |

# 1. I N T R O D U Ç Ã O:

## 1.1 Situação do trabalho:

O desejo de Sistemas de Informação melhores (mais confiáveis, mais baratos, mais funcionais, etc.) é expresso tanto por usuários, como por gerentes e profissionais da área. Grande parte dos projetos continua ultrapassando os orçamentos iniciais e não atendendo os objetivos dos usuários [CAM 87].

Em vários aspectos, produtos de Software, resultando do desenvolvimento de Sistemas de Informação, se parecem com produtos da engenharia convencional como pontes, prédios e computadores. Existem, no entanto, algumas diferenças importantes que dão à Engenharia de Software uma personalidade própria:

- O software é um produto lógico e não um produto físico (mais próximo de uma prova matemática do que da construção de uma ponte),
- O Software não desgasta, sua confiabilidade é determinada por conceitos abstratos como correção, tolerância a falhas, etc...,
- O número de soluções de implementação "corretas" é muito grande,
- Os Sistemas de Informação têm necessidade constante de evoluir ao longo do tempo [LUC 87].

Muitas metodologias têm sido propostas para as diferentes fases do ciclo de vida dos Sistemas de Informação (SI). Inclusive, na área de programação, muitas técnicas, como programação estruturada, programação modular e abstração de dados contribuiram para por fim à era dos programas desestruturados, mas ainda há problemas a serem melhor resolvidos, como os de prazo e de custos de manutenção [GRA 86].

Há uma grande quantidade de metodologias propostas para o desenvolvimento de Sistemas de Informação, ao ponto de ser importante saber qual é a mais adequada, se há uma mais indicada e segundo quais critérios.

Não basta ter uma metodologia se ela não se enquadra em nosso ambiente técnico e humano. Um bom projetista não é um especialista numa metodologia [BOU 87], ele precisa ter muita inteligência e criatividade.

Daí, a idealização pelo **Banco Francês e Brasileiro (BFB)** da macrometodologia **MIND-BFB**, combinação de três metodologias (controle de projeto, especificação de sistema e construção de solução), e este trabalho se enquadra nesse contexto. Baraldi definiu as bases da macrometodologia [BAR1 87] e detalhou a

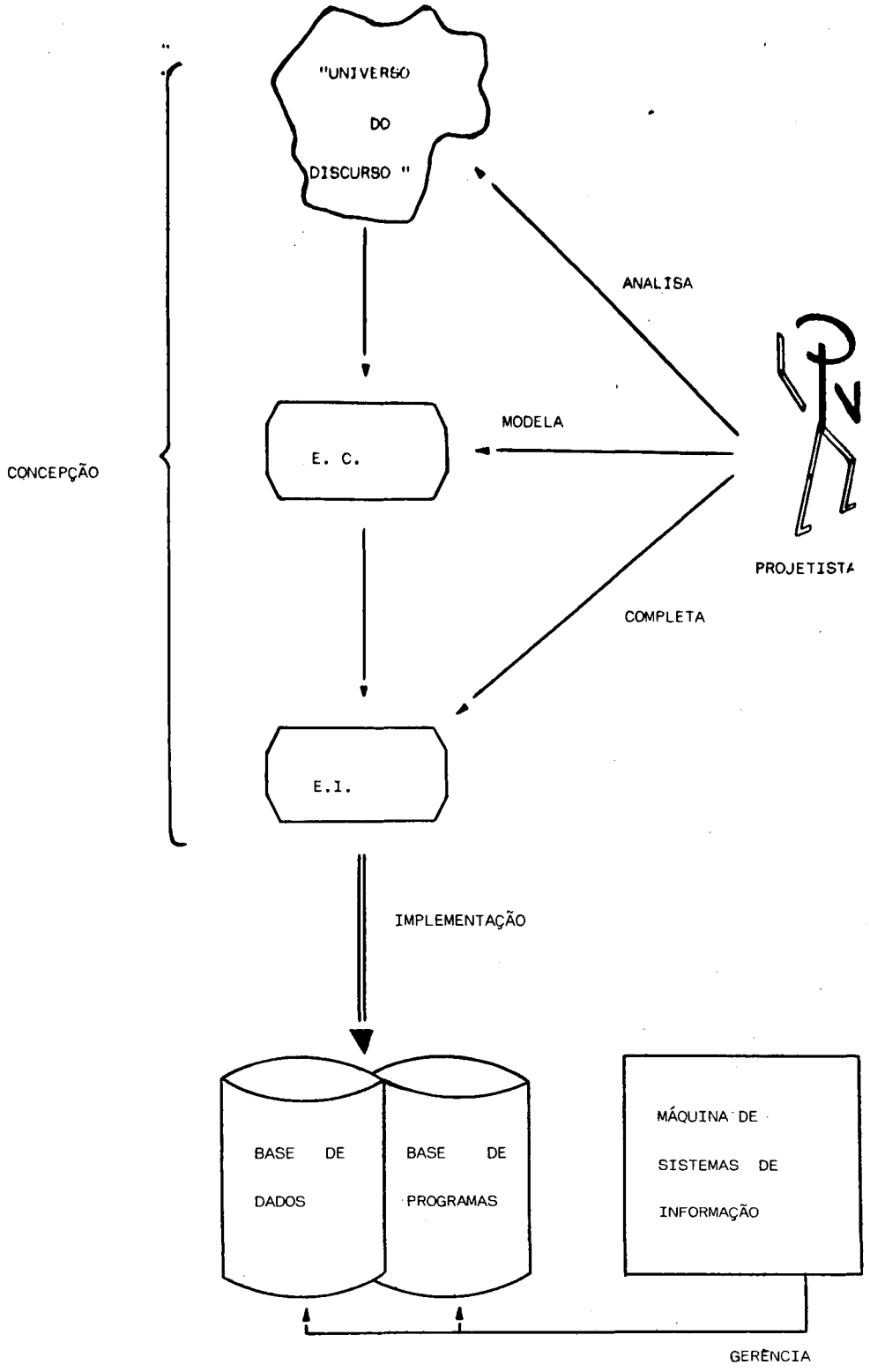



Figura 1: *As diferentes fases da Metodologia REMORA.*

metodologia de especificação de sistemas [BAR2 87] baseada na metodologia **REMORA**, que foi desenvolvida nas universidades de PARIS e NANCY, FRANÇA, pelo grupo de pesquisas da professora COLETTE ROLLAND [ROL 82].

O objetivo deste trabalho é apresentar essa metodologia não muito conhecida no Brasil e propor uma ferramenta de apoio ou "CASE" (Computer-Aided Software Engineering) ao projetista na fase de especificação.

A metodologia REMORA propõe a concepção de SI em duas fases: a conceitual e a interna [ROL 82] e [ROL 79].

A primeira fase tem como objetivo a descrição do sistema real. Seu resultado, chamado ESQUEMA CONCEITUAL (EC), é uma representação formal da estrutura natural dos fatos do mundo real percebidos sob seus aspectos estáticos e dinâmicos. O EC representa simultaneamente a estrutura dos elementos (Sub-Esquema Estático) e a estrutura do comportamento da organização (Sub-Esquema Dinâmico).

A segunda fase parte do resultado da primeira e considera os aspectos técnicos da solução, ignorados nela. REMORA considera igualmente as particularidades da utilizacão. Em cada uma das duas fases, a solução é obtida através de modelagem usando conceitos teóricos e regras metododológicas expressas através de uma linguagem formal **(ver figura 1).**

## 1.2. Metodologias e ferramentas de apoio:

### O que é Sistema de Informação (SI):

Odile Thiery [THI 85] adota a seguinte definição de Sistema de Informação: "Uma combinação formalizada dos recursos humanos e informáticos resultando da coleta, da memorização, da comunicação e do uso dos dados para permitir o gerenciamento eficaz dentro da organização".

O "universo do discurso" (que é a realidade da organização) é projetado sobre o SI com reduções temporais e espaciais [THI 85]. Alguns fatos elementares da organização são considerados e memorizados sob forma de dados. Essa coleção de dados estruturados e de transações forma o SI, que vai artificialmente simular a operação da organização. O SI, idealmente, deverá conter a história da organização e deverá poder nos restituir informações desejadas nas circunstâncias (momento, local, forma) especificadas. Como a própria organização, o SI tem um ciclo de vida: nasce durante sua implementação, evolui com a organização e morre quando se torna obsoleto.

## Necessidade das metodologias de desenvolvimento de SI:

Foi visto que um SI tem um ciclo de vida. A obsolescência de um SI pode decorrer do abandono da atividade da organização (caso normal) mas pode também decorrer da incapacidade do próprio SI de se adaptar à evolução técnológica no decorrer do tempo. Garantir longa vida ao SI é um dos objetivos das metodologias de desenvolvimento de SI.

A partir dos anos 60, a um pedido pontual do usuário, o analista de sistemas automatizava uma atividade da organização, imitando servilmente a forma manual daquela atividade, sem questionar os fundamentos da solução [CAM 87]. Essa atitude resultava numa asfixia do centro de processamento de dados (apesar da precariedade dos recursos técnicos, seus usos não são otimizados) e na insatisfação do usuário (informações desencontradas e raramente disponíveis).

Para evitar o caos, surgiram metodologias para organizar diferentes aspectos do ciclo de vida do SI: da especificação do problema à construção da solução, passando pelo controle do projeto. É bom notar, também, a contribuição das teorias sobre modelagem de dados.

Basicamente, uma metodologia é uma técnica que:

- propõe uma divisão do trabalho
- especifica os problemas a serem tratados
- indica a maneira de colocar e de resolver os problemas [ROL 80].

Pode-se classificar as metodologias de análise em três grandes tendências: a funcional, a estrutural e a de informação [McC 88] e [BER 86].

## A tendência funcional:

Ela dominou as primeiras épocas da análise e continua a ser a mais usada nas organizações. O princípio das metodologias desse grupo é construir um sistema através da análise de suas funções. Soluções técnicas elaboradas realizam essas funções, que são as necessidades imediatas dos futuros usuários. Os resultados fornecidos pelas metodologias são aplicações que só aumentam com o decorrer do tempo (uma nova unidade funcional para cada nova necessidade expressa). Elas facilitam a criação de redundâncias de dados e de procedimentos. Os problemas dos usuários são expressos "orientados para a solução", o que reduz o número de soluções possíveis [UCG 88].

Como ilustração, propomos uma análise crítica de um exemplo desse tipo de metodologia, a mais representativa pela sua popularidade, a Análise Estruturada difundida por CHRIS GANE e TRISH SARSON [GAN 83].

Alguns dos problemas da Análise Estruturada são por exemplo a sua inadequação para abordar problemas de tempo real (ela foi concebida para aplicações relativamente simples) e uma certa pobreza conceitual (tanto o modelo como o formalismo do método), [BOU 82]. Em compensação, para problemas apresentando caracteristicas tais como poucas formas de uso, isolados, usando poucos arquivos e processos, a Análise Estruturada se revela uma metodologia simples e uma boa linguagem de comunicação entre usuários e analistas.

Diante desses problemas, a Análise Estruturada evoluiu para, por exemplo, enfrentar os três aspectos importantes e atuais da complexidade: as **funções**, os **dados** e o **tempo**. Foram propostas novas versões da Análise Estruturada como "Desenvolvimento Estruturado de Sistemas em Tempo Real" de PAUL WARD, [WAR 86] e [BER 86]. Aspectos de modelagem de dados, usando Diagramas Entidade-Relacionamento também foram incorporados.

Se um sistema for complexo em relação a uma dessas dimensões, mas relativamente simples para as duas outras, então as formas clássicas da Análise Estruturada são suficientes. Mas se a aplicação contém funções complexas e aspectos de tempo real, é então nacessário usar as novas versões da Análise Estruturada [YOU 86].

## A tendência Engenharia de Informação:

A Engenharia de Informação originada por James Martin e outros [MAC 86] é uma abordagem relativamente recente que tenta considerar a organização como um todo (entidades, processos, dados e hardware que interagem entre si). A Engenharia de Informação tem tido boa aceitação devido a dois dos seus conceitos principais:

- planificação estratégica: desenha-se o modelo hierárquico da organização e determina-se as informações necessárias para atingir um dado objetivo.

- modelagem de dados: dependências funcionais e associações entre entidades [McC 88].

A Engenharia de Informação começa a um nível mais alto (planificação estratégica) que as tradicionais metodologias da Engenharia de Software. Ela se baseia numa visão lógica de como a organização e o sistema usam os dados e dá mais ênfase às necessidades da organização em relação ao sistema. A Engenharia de Informação preconiza a modelagem dos dados para criação de Bases de Dados normalizadas. Assim, obtém-se um Sistema de Informação integrado. Os resultados são diagramas mostrando

Modelos Entidades-Relacionamentos, Estruturas de Dados, Decomposição Hierárquica das funções, Fluxo de Dados, as dependências funcionais, "layouts" de telas e de relatórios e lógica detalhada dos programas.

Depois desta introdução, pode-se falar na metodologia de James Martin. O objetivo da metodologia ( "James Martin Information Methodologie" ) é gerenciar o desenvolvimento e as interações do sistema através do controlador central de um único modelo de dados. A metodologia considera todos os sistemas da organização para idêntificar como a informação é usada e compartilhada entre eles. O método é descrito em quatro passos:

1) Criação de um plano estratégico: Este plano é criado para todos os sistemas da organização definindo os objetivos a serem atingidos nos cinco ou dez próximos anos e criando modelos conceituais da organização e dos seus dados. O modelo da organização define as funções básicas (como vendas, por exemplo) e a estrutura organizacional usando Estruturas Hierárquicas, Decomposiçaõ de Diagramas e Diagramas de Entidade-Relacionamento. As entidades decorem de entrevistas com usuários (Engenharia de Informação) ou da análise de sistemas já existentes (Engenharia Reversa). Este modelo é o ponto de partida para a modelagem dos dados e para determinar a ordem de desenvolvimento dos sistemas da empresa toda.

2) Modelagem de dados: Este passo usa o resultado do passo 1, extraindo do modelo de Entidades-Relacionamentos global a parte que interessa a uma aplicação dada para normaliza-la. Por exemplo, do modelo relativo a "vendas" pode ser considerado apenas a parte relativa a fornecimento. Aqui serão então definidas as entidades, os atributos, os relacionamentos entre as entidades e os sistemas a serem construídos para áreas delimitadas. O Diagrama de Decomposição Funcional representando as grandes funções é detalhado para obter-se processos agrupados por àrea. Uma matriz de referência cruzada mostra que processo usa que entidade.

3) Especificação dos processos: Depois de escolher os processos a serem construídos para uma aplicação, dependendo do caso, o processo pode existir ou pode ser comprado como pacote ou então pode ser da responsabilidade do Centro de Informação que dispõe de um ambiente de Quarta Geração. Caso contrário, o desenvolvimento continua com a especificação dos algoritmos e das estruturas de dados e dos processos. Poderá ser usado aqui a Análise Estruturada, por exemplo, para determinar DFDs, "layouts" de telas, de relatórios, os algoritmos.

4) Implementação: Quando o processo não teve que ser especificado detalhadamente (estruturas de dados, algoritmos, layouts), este passo consiste a implementar fisicamente as estruturas de dados e os algoritmos numa linguagem dada. Assim, a aplicação é entregue com as especificações lógicas e físicas.

## A tendência estrutural:

Essa tendência coresponde ao desenvolvimento das bases de dados e dos gerenciadores de bases de dados.

O objetivo é construir um sistema através da definição da estrutura de seus dados. Várias proposições sugerem a introdução do nível conceitual dos dados no processo de concepção de SI [ROL 82]. A abordagem enfatiza a representação semântica dos fatos do SI. Ela fornece ferramentas formais para validar as qualidades da representação dos dados (eliminação das redundâncias, consistência, etc...) e permite uma verdadeira independência entre os problemas e as soluções técnicas. Para essas metodologias, o essencial reside na compreensão do SI como sistema pertencente ao conjunto de todos os sistemas da organização. Elas procedem a uma decomposição rigorosa do SI em elementos e à definição das relações entre os elementos.

A metodologia **REMORA** pertence a essa família, pois ela foi concebida com essa visão e se preocupa em fornecer um modelo da realidade organizacional baseada na memorização de todos os fenômenos pertinentes da organização [ROL 88] e [ROL 79].

## Ferramentas de Apoio ou "CASEs":

Para dar apoio ao uso de metodologias, muitas delas concebidas originalmente para serem aplicadas de forma manual, surgiram softwares desenvolvidos para esse fim: são as chamadas "Ferramentas de Apoio", ou CASEs (Computer-Aided Software Engineering).

Essa "Engenharia de Software" com apoio de computador está hoje no seu início. Prevê-se que em 1990, 10% dos programadores terão acesso a ferramentas de apoio e que, em 1995, serão 90% [YOU 86].

Outra expectativa é que as metodologias de análise sejam combinadas com as técnicas de Inteligência Artificial para produzir Sistemas Especialistas que permitam a interação em linguagem natural com os usuários [BOU 87] e [LUC 87].

## Tipos de CASES:

Um CASE pode ser caracterizado basicamente por duas de suas dimensões: as metodologias em torno das quais ele é construído e as fases do ciclo de vida do sistema que ele abrange. Estes dois fatores são ligados. No caso da metodologia, o CASE deve conhecer e exigir o uso adequado dos conceitos, da linguagem e do método desta para levar o usuário a uma especificação coerente [McC 88] e [DAR 87].

No caso da fase do ciclo de vida, o CASE deve assistir em todas as tarefas necessárias naquela fase e levar a um resultado que seja completo e coerente para ser usado como entrada da fase seguinte. É lembrado a seguir ,as fases do ciclo de vida de sistemas utilizado tradicionalmente:

. Planejamento: coleta de informações sobre os problemas e as necessidades do usuário, definição de objetivos e critérios e geração de soluções alternativas.

. Análise: determinação das necessidades do usuário e das restrições do sistema, confrontação das soluções com as necessidades expressas, geração de especificação funcional e modelo lógico para a solução retida.

. Projeto: detalhadamento da concepção para a solução selecionada (fluxos, programas, funções, dados).

. Implementação: implementação e testes.

. Manutenção: modificações e correções ao decorrer do tempo.

Uma maneira de classificar as metodologias (para as quais os CASEs dão apoio) é sua orientação que pode ser para dados, processos ou informação:

. orientados_a__dados: As estruturas de dados (entradas e saídas) são definidas com prioridade. Daí são derivados as estruturas dos processos.

. orientados_a_processos: Os processos formam a parte fundamental da modelagem e para isso são os primeiros analisados. Depois, os fluxos de dados são definidos em torno dos processos.

. orientados_a_informação: Começam com uma modelagem de dados representando o uso da informação dentro da organização. O projetista analisa a organização, seus objetivos de negócio e sua estratégia de informação.

No estágio atual da tecnologia de CASEs, existem cinco tipos de CASEs, que são [McC 88]:

.CASEs__individuais ("individual CASEs"): automatizam um aspecto de uma fase do ciclo de vida de uma aplicação. São ferramentas para diagramação, layouts, dicionários, verificação de especificação, geração de código, documentação, etc...

.Kits_de__ferramentas ("toolkits"): ferramentas integradas para uma fase do ciclo de vida de sistemas. São ferramentas para análise (prototipação, dicionário,especificação), concepção de dados (modelagem de dados, normalização, geração de estruturas), programação (controle de sintaxe, edição, geração de código), gerenciamento de projeto ( processamento de texto, calendário de tarefas, tabelas de estimação, métricas).

.Bancadas ("workbenchs"): ferramentas integradas que automatizam todo o desenvolvimento e atualização de sistemas, que ajudam no gerenciamento do projeto. A saída de uma fase do ciclo de vida é direta e automaticamente passado para a próxima fase. A saída final é um sistema implementável, executável e documentado.

.Integradores ("frameworks"): juntam, gerenciam e facilitam o uso de ferramentas isoladas de autorias diferentes, antigos ou novos. Contribuem para tranportabilidade destas ferramentas de uma configuração para outra.

.CASEs de Apoio a Metodologias ("methodologie companions"): são CASEs centralizados em torno de uma metodologia específica obrigando a usar conceitos e métodos daquela.

A seguir uma análise de duas dessas ferramentas. Ênfase maior é dada à segunda, que o autor chegou a estudar na prática.

**Action Diagrams:**

O "Action Diagrams" é uma ferramenta baseada nos conceitos da programação estruturada [RUS 86]. Ela ajuda nas duas abordagens macro e micro da especificação de sistemas. As técnicas como LCP [WAN 80] e JSP [CAR 86] são usadas pelos analistas para entender e conceber as estruturas gerais e globais dos programas enquanto que as técnicas como Português Estruturado e diagramas de Nassi-Sheiderman são usadas para especificar programas em nível de detalhes.

A passagem de um a outro desses níveis pode ocasionar erros, já que são técnicas diferentes, com conceitos diferentes. Daí a proposta de uma ferramenta que assista o projetista em todos os níveis de detalhadamento de programas e de maneira integrada.

Um processo é especificado dentro de um colchete ("[") automaticamente criado para formar um bloco de instruções. Exemplo:

```
----
!  ler  Preco-Unitário
!  ler  Quantidade
!  imprimir "total = ",Preco-Unitário * Quantidade
----
```

O bloco não tem limitação no tamanho. Dentro de um bloco, podem ser criados outros blocos.

Para representar iteração, usa-se colchete com duas horizontais no topo:

```
==== para todos os itens
!  calcular o preço total
----
```

As estruturas de controle condicionais (SE ENTÃO, SENÃO e CASO) são construídas com um colchete, tendo uma horizontal para cada caso.

```
 ---- SE D < 1
|  ler  P
 ---- senão
|  ler  P e Q
 ----
```

Uma instrução de um bloco pode ser detalhada num outro bloco de nível inferior (especificação "TOP DOWN"). É assim, que o primeiro exemplo pode ser considerado um detalhamento do segundo.

A ferramenta controla o uso da programação estruturada. Além de gerir as dependências de níveis ("pais" e "filhos" são associados e podemos percorrer a especificação de baixo para cima e vice-versa), ela disciplina o usuário no uso das estruturas de controle.

A tabulação do texto é automática durante a especificação. Quando se chega a um nível de especificação bastante detalhado, pode-se obter automaticamente a conversão do texto numa linguagem de programação de alto nível.

**Structured Architect:**

O STRUCTURED-ARCHITECT é um CASE da análise estruturada, que permite:

- editar diagramas DFD
- criar uma dicionário para processos e entidades externas
- criar uma dicionário para dados (fluxos e depósitos)
- obter análise de "anormalidades" dos diagramas
- emitir relatórios

Ele oferece facilidades como ZOOM, aceleração do cursor, painel de comandos, etc. Em todos os níveis podemos recorrer ao comando HELP em caso de dúvidas ou ESCAPE para desfazer uma tarefa.

O Sistema oferece várias facilidades durante a edição:

- passagem de um diagrama para processo "pai" que foi decomposto.
- passagem de um processo para a sua decomposição.
- possibilidade de usar o zoom para expandir ou contrair o diagrama na tela.
- possibilidade de regular a velocidade do cursor.
- um painel (opcional) a direita da tela para informar as ações possíveis.

A seguir será feita uma breve exposição de cada um dos componentes deste software:

a) O editor de DFD permite:

- criar ou remover objetos (processo, depósito, fluxo, entidade externa )
- renomear objetos
- movimentar objetos.

Para cada um desses objetos, o Structured Architect pede para o usuário colocar o cursor sobre o objeto e teclar <ENTER>, o que lhe permite identificar o objeto. Também pode ser usado o "mouse". No caso da renomeação, o CASE verifica se não vai haver duplicação de nomes ja existentes, a duplicação é permitida para fluxos de dados, entidades externas e depósitos de dados (conforme [GAN 83]) mas não é permitido para processos.

Se for remover um objeto, o sistema pede confirmação se deseja remover todas as informações associadas ao objeto. Na movimentação de um processo, de um depósito ou de uma entidade externa, todos os fluxos ligados são movimentados também.

b) Enciclopédia:

A enciclopédia é composta de um conjunto de informações relativas a processos, entidades externas, fluxos de dados e depósitos de dados:

- Dicionário dando a lista alfabética desses tipos de objetos e especificando se é entidade externa, processo ou primitiva funcional (processo não decomposto).
- Fluxo de Dados associados a cada um dos dois tipos de objetos objetos, separando aqueles que entram daqueles que saem.
- O Português Estruturado relativo a cada primitiva funcional.
- A descrição de cada um dos objetos (comentários livres).
- A estrutura de cada fluxo ou depósito de dados.
- Uma descrição sumária do objeto pode ser também criada e mantida.

A notação usada nos manuais de programação para especificar as características de uma estrutura de dados são também utilizadas para representar as estruturas de dados:

[ ] para estruturas de dados opcionais
{ } para estruturas de dados alternativas
* para estruturas de dados iterativas

c) Relatórios de análise:

O sistema faz uma análise dos elementos criados e produz para o analista vários tipos de análises de consistência, mostrando aspectos variados da especificação:

- balanceamento de um processo com seu diagrama (decomposição), dando a lista dos fluxos que entram ou saem do processo e que não

são representados no diagrama correspondente e vice-versa:

- A lista dos fluxos que entram/saem em processos e que não são representados nos diagramas correspondentes.

- A lista dos fluxos que entram/saem de diagramas e que não são representados no processo "pai".

- A lista dos objetos que ainda têm informações associadas nas enciclopédias, mas que deixaram de existir no diagrama (remoção).

- A lista dos objetos sem descrição.
- Primitivas funcionais sem português-estruturado.
- Elementos de dados sem "PICTURE".

d) Relatórios gerais:

O sistema gera vários relatórios na tela ou na impressora:

- lista de todos os objetos
- estrutura dos processos
- estrutura dos dados
- o diagrama de fluxo de dados (ver figura 2), etc.

## 1.3. Organização da dissertação:

Neste capítulo, é apresentado o tema do trabalho e é feita uma apresentação das metodologias e dos CASEs. No fim, é comentada a organização do trabalho.

O segundo capítulo contém a análise da bibliografia básica utilizada no trabalho. O primeiro conjunto de documentos apresentados trata da metodologia REMORA, o segundo grupo contém publicações sobre CASEs e, em fim, dois artigos de abordagem geral são comentados.

O terceiro capítulo introduz a metodologia REMORA, mostrando o interesse do modelo geral escolhido (que é uma visão mais ampla que a das Bases de Dados) para depois apresentar o modelo descritivo, o modelo conceitual e o nível interno. O capítulo termina com um resumo da macrometodologia que considera REMORA como um de seus componentes e um exemplo.

O quarto capítulo descreve a ferramenta de apoio proposta para a metodologia REMORA, que será denominada MIND-AID. Em primeiro lugar é mostrado a abrangência da ferramenta e suas funções globais, depois uma descrição mais detalhada apresenta os diferentes módulos para em fim comentar um pouco a linguagem usada para a implementação da ferramenta.

O quinto e último capítulo contém as conclusões gerais do trabalho, assim como uma avaliação da ferramenta desenvovida.

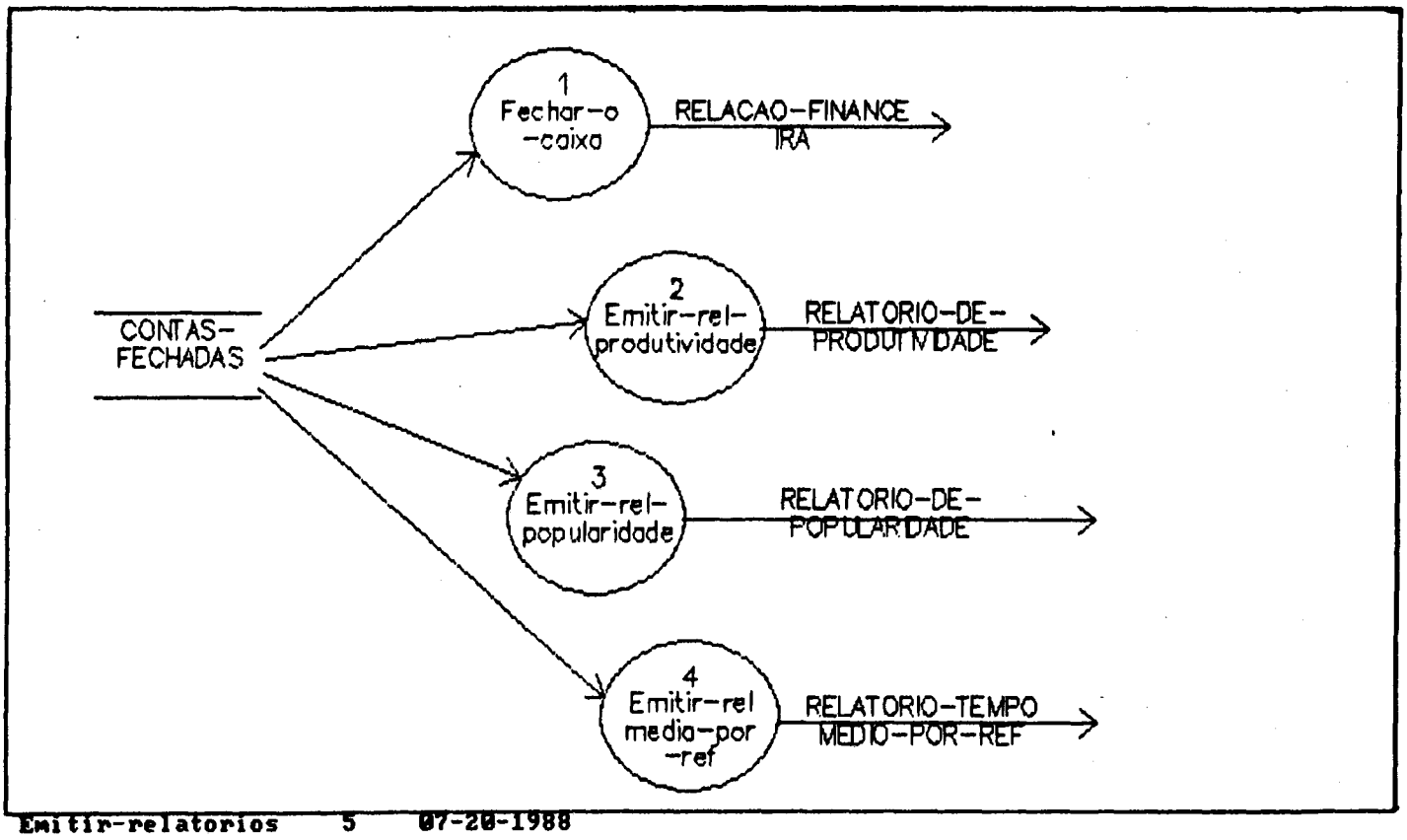



Figura 2: *Um Diagrama de Fluxos de Dados emitido pelo CASE: "Structured Architect".*

## 1.4. Conclusão:

Os Sistemas de Informação são, hoje, de capital importância para as organizações. A competitividade entre elas dependerá bastante das qualidades do SI que possuem. Essas qualidades que são completude, disponibilidade e coerência das informações da organização gerida pelo SI dependem, por sua vez, da metodologia empregada para conceber o SI.

Uma das razões que explica a não utilização das metodologias como pode-se esperar é a quantidade de trabalho na hora da concepção, a facilidade de cometer erros (em relação ao modelo da metodologia empregada) e a impraticidade de manter atualizada a documentação em caso de modificações. Para isso se faz necessário dispor de ferramentas automatizadas para facilitar o emprego das metodologias. Surgiram várias recentemente e foram apresentadas duas delas neste trabalho.

Apesar do grande otimismo com que alguns vêem a primeira (Action Diagrams) [RUS 86], essa ferramenta pode ser de grande ajuda em certas situações tais como: sistemas pequenos, isolados, usando poucos dados, trechos da especificação de uma solução, etc.. Contudo, dificilmente conseguirá enfrentar projetos industriais com dados de grande volume ou complexos. Por isso, ele faz parte de uma ferramenta mais abrangente, o IEW da KNOWLEDGEWARE [YAY2 88].

Quanto à segunda (Structured Architect), foi visto que ela oferece bastante apoio ao analista no nível lógico da análise mas não chega a auxiliar na fase de projetos. Além disso, ela deixa a desejar quanto às averiguações de consistência e integridade, por exemplo:

- Pode-se usar um elemento de dado no português estruturado e o Structured Architect não verifica se esse elemento veio através de um fluxo de dados ou depósito do DFD.

- Pode-se definir um elemento num depósito de dados e o Structured Architect não verifica se esse elemento veio de um processo, de fluxo externo ou de um outro depósito.

- Quando se proceder à expansão de um processo ao invés de exigir a representação dos fluxos externos, chama a atenção do usuário sobre a "anomalia" que consiste em repetir o mesmo nome de fluxo.

Finalmente, o STRUCTURED-ARCHITECT é util para documentação mas apresenta pouco formalismo e rigor. Para isso o analista usuário terá que estar atento para garantir a coerência e a integridade do seu trabalho.

Como qualquer nova tecnologia, a dos CASEs tem muito a fazer ainda para atingir os objetivos que ela tenta atingir. Em [YAY2 88], se encontra o resultado de um estudo que o autor juntamente com outro colega, fizeram sobre os CASEs.

## 2. Bibliografia básica e trabalhos correlatos:

### 2.1. Introdução:

Neste capítulo, serão apresentadas a bibliografia básica do trabalho. Em três itens, serão apresentados sucessivamente artigos, livros e outras publicações importantes sobre a metodologia REMORA, os CASEs e outros assuntos relacionados com o trabalho. Para a metodologia REMORA, foi conseguida uma coleção de teses e artigos. Quanto aos CASEs, alguns artigos publicados recentemente ajudaram a tratar do problema e situa-lo de forma mais clara.

### 2.2. Publicações sobre a metodologia REMORA:

Num primeiro artigo, C. Rolland, G. Benci e S. Leifert apresentam um método de concepção de Bases De Dados que vai corresponder mais tarde àquele utilizado na concepção do Sub-Esquema estático de REMORA. O artigo detalha como estruturar a realidade da organização para chegar a uma representação da estrutura conceitual [ROL 80]. É bom observar que a esse nível não se falava ainda das Formas Normais Permanentes.

Para a conferência da IFIP em 1982, foi publicado o primeiro documento sobre REMORA, para ser avaliado junto com várias outras abordagens. Cada uma das metodologias deveria apresentar uma solução para o problema de organização de uma conferência. Os autores apresentaram REMORA de maneira sucinta mas completa. Esse artigo apresenta didaticamente a diferença entre a tendencia funcional e a tendencia estrutural, a Terceira Forma Normal Permanente e o exemplo da conferência é desenvolvido completamente. É um dos poucos documentos disponíveis que trata o nível lógico de REMORA [ROL 82].

Um ano depois, C. Rolland, C. Richard e S. Leifert explicam a os quatro componentes de uma metodologia que são MODELO, LINGUAGEM, MÉTODO e FERRAMENTAS. Mostram no mesmo documento uma definição integrada das estruturas de dados, dos processos e da sincronização dos comandos. A linguagem ISDEL é apresentada, assim como o modelo de REMORA [ROL 83].

Na primeira parte da sua tese de doutorado, em 1982, Odile Foucaut faz uma análise da semântica dos dados, dos programas, e dos comandos dinâmicos de um SI. Ela mostra que certos aspectos da realidade são mal representados nos SI atuais (sobretudo o aspecto dinâmico). Na segunda parte, são apresentados o modelo descritivo e o modelo conceitual de REMORA. Finalmente, na terceira parte, ela estuda o tipo de modelagem resultante do uso do modelo e apresenta a Máquina de Sistema de Informação (MSI) que é a descrição de um automato abstrato para assistir um projetista [FOU 82].

Uma outra tese de doutoramento sobre REMORA foi publicada em 1985 por Odile THierry. Ela apresenta uma análise bibliográfica de métodos de análise e chega à conclusão que seria vantajoso a introdução dos tipos abstratos de dados em REMORA. Em seguida, ela desenvolve um formalismo permitindo expressar os conceitos de REMORA. A terceira parte consta da apresentação de uma linguagem de especificação, LASSIF, considerando o conceito de abstração de dados. LASSIF é também o nome de uma ferramenta de apoio à concepção que ela propõe para a gestão do SI no descurso do tempo [THI 85].

Em 1987, Baraldi publicou um documento interno ao BANCO FRANCÊS E BRASILEIRO justificando e apresentando a macrometodologia MIND-BFB [BAR1 87]. Com base nesse documento, foi posteriormente detalhada a metodologia de especificação de sistemas dentro do contexto dessa macrometodologia [BAR2 87].

Recentemente Foi publicado um livro pelo grupo de pesquisa de C. ROLLAND. O livro é uma exposição didática da metodologia, fazendo-se abstração de alguns detalhes e justificativas teóricas. Ele se parece mais como um resumo de [FOU 82], completado com vários exemplos ilustrando os conceitos [ROL 88].

## 2.3. Publicações sobre "CASEs":

Como bibliografia sobre CASEs, pode-se citar, novamente, a tese de Odile Thierry, [THI 85], onde a arquitetura básica, exemplos de resultados e listagens da ferramenta LASSIF são apresentados.

Para a mesma metodologia REMORA, uma ferramenta de apoio está em desenvolvimento na França, pelo grupo de pesquisa da C. ROLLAND. O artigo [PRO 88], fonte dessa informação, é a descrição de um Sistema Especialista (SE) para a concepção de um SI. É mostrado o interesse de um SE na concepção de SI antes da descrição do OICSI (Outil Interactif de Conception de Systemes d'Information) que é uma ilustração dessa idéia.

OICSI é uma ferramenta para a metodologia REMORA (CASE de Apoio a Metodologia), que gera o Esquema Conceitual do sistema a partir de uma especificação em "linguagem natural".

A concepção dos SI é caracterizada por dois tipos de conhecimentos:

- o conhecimento do domínio da aplicação detido pelos profissionais, usuários potenciais, e que têm desse conhecimento uma visão pragmática, pontual e detalhada e não um modelo global e sintético;

- o conhecimento necessário para a concepção, que são conceitos formais e algoritmos (ex: normalização) mas que geralmente dependem da experiência e de métodos heurísticos.

Como essas duas formas de conhecimento são informais e/ou de tipo experimental, o problema de concepção de SI tem mais razão de pertencer ao domínio da Inteligência Artificial (IA) do que ao do processamento algoritmico.

Foi então escolhida a implementação de um SE para assistir à concepção de SI seguindo a metodologia REMORA. As vantagens dessa escolha são:

- a integração na mesma Base de Conhecimento dos conceitos e algoritmos desenvolvidos na teoria relacional e das regras específicas e experimentais decorrentes da experiência do projetista;

- a amplificação das capacidades de resolução do projetista da mesma forma que o SI amplifica as capacidades de memorização dos profissionais;

- o fornecimento de um ambiente interativo que ajuda e controla a concepção;

- a possibilidade de evolução da própria ferramenta devido à natureza "open system" dos SE.

A cada um dos três níveis tradicionais de concepção de SI corresponde um tipo de conhecimento:

- modelagem descritiva: conhecimento de tipo heurístico dependente do projetista;

- modelagem conceitual: conhecimento mais formal baseada nos conceitos da metodologia usada e nas regras que tratam da sua utilização;

- modelagem interna: conhecimento formal das regras de transformação de esquemas e conhecimento experimental do tratamento das regras de otimização.

O SE deve, então, incluir o conjunto dos conhecimentos (formais e heurísticos) que são usados pelo projetista durante a concepção.

O objetivo do OICSI é recolher um conhecimento suficiente do domínio da aplicação e produzir progressivamente, em interação com o projetista, os esquemas conceitual e lógico da metodologia REMORA.

Nas organizações, geralmente não existe documento que expressa todo o conhecimento necessário à concepção. Ele está na cabeça dos profissionais. Essas pessoas transmitem esse conhecimento mais através de palavras que de documentos. Para isso OICSI contém um interface permitindo que o conhecimento seja coletado através uma lingua natural: o francês.

Mas a riqueza da lingua natural conduz a uma análise e uma interpretação complexas. Para reduzir essas dificuldades, foi escolhido um sub-conjunto do francês.

Um estudo sistemático dos documentos de concepção de SI em francês permitiu a definição de um sub-conjunto, que é suficiente para descrever a realidade. Exemplos de restrições na lingua são:

- as frases devem estar na terceira pessoa do singular ou do plural, no presente do modo indicativo;

- as frases devem ser o mais independentes possíveis umas das outras;

- não inversão do grupo sujeito em relação ao grupo verbal;

- não utilização de metaforas.

O sistema procede a uma verificação sintática e semântica das frases antes de gerar o esquema conceitual correspondente. Ver, na **figura 3**, a estrutura sintática em árvore de uma frase.

As publicações analisando CASEs são bastante variadas. A seguir as mais importantes:

Num artigo bastante esclarecedor [McC 88], da uma classificação global dos CASEs existentes baseando-se nas fases do ciclo de vida assistida ou nas metodologias apoiadas pela ferramenta. Ela identifica os cinco tipos de CASEs que foram definidos no item 1.2. (CASEs Individuais, Kits de Ferramentas, Bancadas, Integradores e CASEs de Apoio a Metodologias).

Um artigo na mesma linha é o de DART, [DAR 87]. É dada uma classificação dos CASEs em quatro tipos. Dois desses tipos se encontram na classificação da [McC 88]. A classificação adotada pelo autor é a seguinte:

Ferramenta de Apoio a Linguagem ("Language-Centered Environments"): suporte a uma linguagem particular, é um CASE altamente interativo permitindo codificações, testes, correções e execuções rápidas de programas pequenos.

Ferramenta de Apoio a Estrutura ("Structured-Oriented Environment"): inicialmente destinado a fornecer ao programador um editor orientado para a sintaxe de uma linguagem, hoje é um CASE que além de ser uma generalização da Ferramenta de Apoio a Linguagem, dá suporte para gestão de informações como "logs" e controles de acesso. Também pode extrair e fornecer uma interpretação semântica do programa. É capaz também de mostrar um programa a níveis diferentes de detalhes.

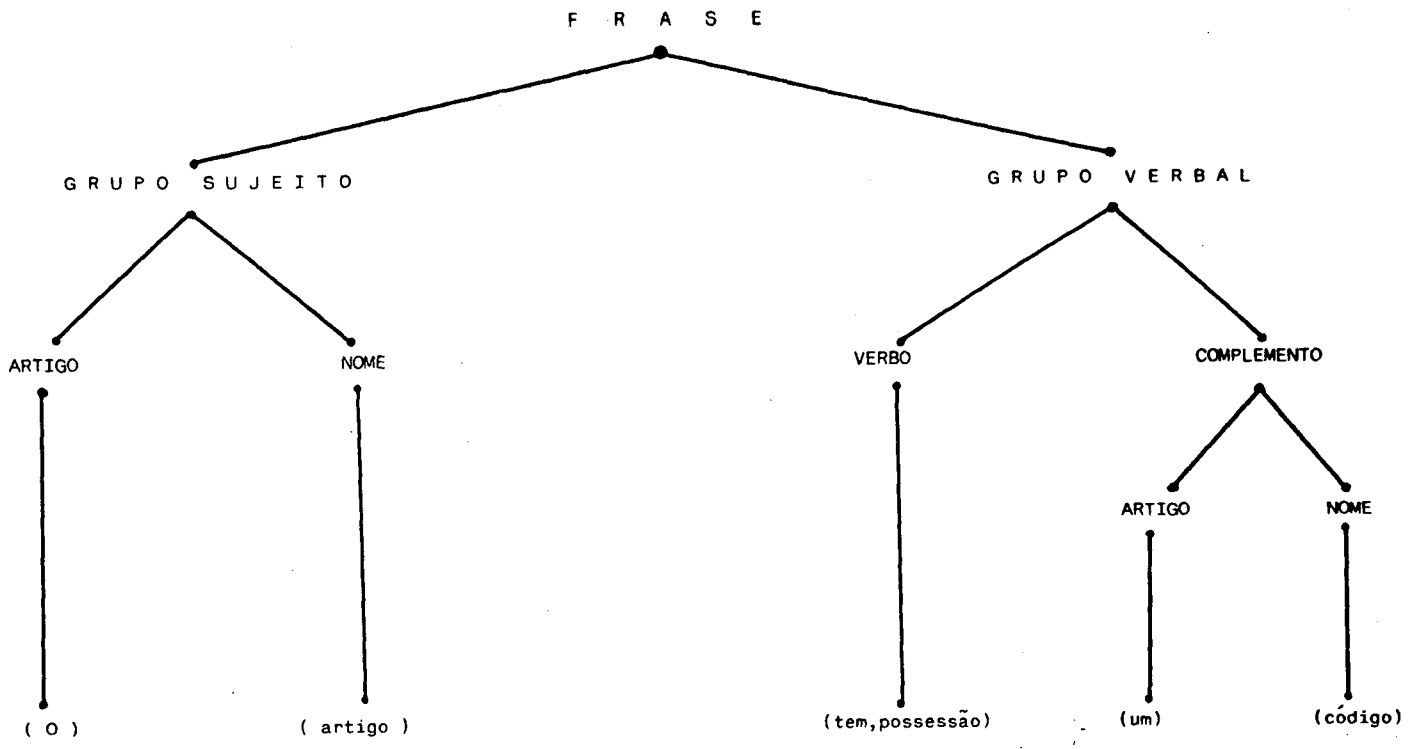



**Figura 3:** *Representação em árvore da estrutura sintática de uma frase analisada pelo CASE: "OICSI".*

Kit de Ferramentas ("Toolkit Environment"): Coleção de pequenas ferramentas para dar apoio na fase de codificação. Usa facilidades do sistema operacional para integrar ferramentas de codificação como compilador, editor, linkeditor, etc. Ele tenta dar apoio a programação em grande escala, independentemente da linguagem.

Ferramenta de Apoio a Metodologia ("Metod-Based Environment"): Dá apoio a uma metodologia específica.

Uma abordagem menos comum é feita por Charlie Bachman, [BAC 88], onde ele defende a necessidade de CASEs e metodologias que suportem as aplicações existentes para incorpora-las nos novos sistemas a serem desenvolvidos: a Engenharia de Software Reversa.

Gibson, [MIC 88], da uma lista de vinte e dois critérios a serem considerados na hora de escolher um CASE para uma organização. Entre esses critérios, alguns dos mais importantes são: metodologia suportada, existencia ou não de um dicionário de dados, suporte a sistemas já existentes, qualidade dos relatórios emitidos pelo CASE, tendencias de evolução, etc. A tabela 1, extraida de [YAY2 88], completa estes critérios e faz uma comparação de alguns CASEs comercializados no mercado nacional.

O artigo de Alda Campos, [CAM 87], analisa a crise do software, como as metodologias tentarão solucionar esta crise e a necessidade das ferramentas para tornar prático o uso daquelas.

## 2.4. Outras publicações:

Dois outros artigos foram de muito grande interesse neste trabalho. O primeiro é o de Yau e Tsai, [YAU 86]. Neste artigo, os autores dão uma visão geral sobre a Engenharia de Software (metodologias, métricas, etc...). É dada, também, uma classificação das metodologias (orientadas a dados e orientadas a processos).

O segundo artigo é de Edward Yourdon, [YOU 86], que analisa a atual crise da Análise Estruturada, apesar desta ter conhecido um momento de grande euforia no meio dos profissionais do processamento de dados na década passada. Para Yourdon, as razões da crise da Analise Estruturada são a impraticidade de manter atualizada a especificação manual, o formalismo limitado do método que leva facilmente a cometer erros e, finalmente, a concorrência de ferramentas de Quarta Geração. O autor termina sugerindo os CASEs como solução para essa crise.

## 2.5. Conclusão:

Apesar de não ser exaustiva, a bibliografia que foi conseguida sobre a metodologia REMORA pode ser considerada importante e bastante representativa. As publicações foram trabalhos profundos (do conjunto fazem parte três teses) e estendem-se do início do

**TABELA 1:** Análise Comparativa de alguns CASEs do mercado Nacional

| ÍTENS | A/D Toolkit | SMART | IEW |
|---|---|---|---|
| Preço do produto básico. | 700 OTNs. | | $ 8 000,00 + 650 por módulo |
| Instalação requerida. | PC-XT,-AT e COMPATÍVEIS | PC-XT,-AT (PC-DOS) e Main Frame (CMS) | PC-AT e MAIN FRAME (MVS) |
| Metodologia suportada. | Análise estruturada(Yourdon) e projeto estruturado, MER. | Análise estruturada. MER | Análise Estr, Modelagem ER, Engenharia de Informação. |
| Diagramas disponíveis. | Diagrama de contexto, ER Tempo Real, Hierárquico, Fluxo de Dados, Fluxogra, etc. | Diagrama de contexto, de fluxo de dados, entidade--relacionamento, de composição, transição e de estrutura (hierárquico). | Diagrama de decomposição, entidade-relacionamento, de fluxo de dados, de ação, de estrutura e matriz de dados x processos. |
| A ferramenta possui um gerenciador de dados ou de dicionário de dados. | Há um repositório central de dados, cujos dados podem ser manuseados. | Há um repositório central de dados, cujos dados podem ser manuseados. | Há uma enciclopédia de dados, que contém e mais do que um dicionário. |
| Especificação do sistema. | A especificação é feita diretamente no diagrama DFD. | A especificação é feita na forma de texto, através de português estruturado. | Diversas linguagens gráficas poderosas. |
| As tendências de evolução da ferramenta. | Será aprimorado o repositório central e sua interface com os diagramas. | | Adaptação para outros sistemas operacionais e outras metodologias. |
| Facilidades de navegação entre os elementos da ferramenta. | A navegação existe, mas não é muito eficiente; um diagrama "pai" não chama o "filho" | | Integração, reflexão e flexibilidades ótimas. |
| A ferramenta cria especificações para sistemas já desenvolvidos, ou seja, dá suporte a sistemas antigos | Não há nenhum suporte a sistemas já construídos, ou seja, não há nenhum apoio à manutenção. | Não dá nenhum suporte à manutenão de sistemas previamente desenvolvidos | Desde o planejamento, possibilidade de integrar o existente. |
| A ferramenta possui interface com outras ferramentas CASE, para que várias metodologias sejam usadas. | Não há nenhuma interface com outros CASEs. | Não há nenhuma interface com outros CASEs. | Import/Export de elementos da enciclopédia. |
| Número de níveis de detalhamento ("explosão") dos diagramas. | Não há limite de níveis na especificação dos diagramas. | Não há limite de níveis na especificação dos diagramas. | Ilimitado. |

| ÍTENS | A/D Toolkit | SMART | IEW |
|---|---|---|---|
| Janelas que mostram simultaneamente diversos diamas. | Até 7 janelas podem estar abertas simultaneamente, mas a navegação entre elas não é muito ágil | | Até 6 janelas podem estar abertas simultaneamente, com reflexão no instante |
| O diagrama corporativo fornece uma boa compreensão ou representação do planejamento da empresa. | Existe um diagrama de contexto que mostra a relação do sistema com o restante da corporação. | Existe um diagrama de contexto que mostra a relação do sistema com o restante da corporação. | Sim, já que dá suporte à Engenharia de Informação |
| A ferramenta suporta a prototipação de sistemas. | Não existe nenhum suporte específico à prototipação. | Não existe nenhum suporte específico à prototipação. | Sim |
| Futuramente a ferramenta transformará automaticate o nível conceitual em nível físico. | Atualmente, não há nenhuma intenção e nem previsão desta automação. | | Já se faz Cobol e CICS. |
| Verificação de erros. | Há verificação de erros a nível de diagramas e de projeto. | Erros de validação e inconsistências são detectados. Nivelamento e balanceamento são feitos automaticamente. | Total |
| Qualidade de emissão de relatórios. | O conteúdo dos relatórios ainda é bem simples,contendo algumas referências cruzadas | Há a emissão de inúmeras referências cruzadas além dos elementos do sistema. | Perfeita |
| Geração de Código. | Não há nenhum suporte na geração de código. | Há geração de uma estrutura dos processos,que deve ser trabalhada para funcionar. | Há geração de código em COBOL, CICS inclusive com comentários. |
| A ferramenta permite distribuição das responsabilidades de desenvolvimento | Não é possível desenvolver o sistema em partes ou módulos e depois reintegrá-los. | Não é possível desenvolver o sistema em partes ou módulos e depois reintegrá-los. | É possível destribuir o trabalho,desenvolve-lo isoladamente e depois integrá-lo. |
| Capacidade de exportar partes de diagramas e de especificações do dicionário de dados. | É possível transferir ou copiar diagramas, especificações e trechos destes. | | É possível transferir ou copiar diagramas, especificações, trechos destes e de qualquer texto especificado |
| Interface com o banco de dados corporativo. | Há comunicação apenas com os dicionários de dados externos através do 'Extract'. | Há saídas para o dicionário de dados corporativo. | Importações e exportações do dicionário de dados corporativo podem ser feitas. |

| ÍTENS | A/D Toolkit | SMART | IEW |
|---|---|---|---|
| Processamento de textos. | Não | Não | Sim |
| Administração de Projetos | Não há módulo que implemente funções de administração e planejamento de projetos. | O módulo 'METRICS' fornece dados para prever esforço e duração do projeto | Sim |
| Possibilidade de adaptação da Ferramenta a outras metodologias. | Existe o módulo 'Rule Tool' que possibilita a definição de outros diagramas e regras | Existe o módulo 'TIE' que permite a definição de apenas outras regras. | Sob iniciativa do fabricante |
| Facilidade de manutenção do projeto e do sistema. | Qualquer alteração em uma especificação não é automaticamente refletida nas demais, há uma indicação do local onde deve ser modificado. | Qualquer alteração feita na especificação do sistema é refletida nos diagramas e no repositório central. | Total |
| Integração entre as ferramentas. | A integração entre as ferramentas é pequena,pois uma especificação não gera outras especificações. | Há uma boa integração, pois de uma especificação são gerados automaticamente diversos diagramas. | Total |
| Nível de Inteligência associada. | A Inteligência Artificial é utilizada para a verificação de erros. | Linguagem de IA é utilizada para definição de regras de produtividade e planejamento | IA é utilizada para verificação de erros,definição de regras,construção de diagramas |
| Tipo de Case | Individual Case de Diagramação. | Toolkit de Análise | É um Workbench da Engenharia de Informação de James M. |

| ÍTENS | A/D Toolkit | SMART | IEW |
|---|---|---|---|
| Ocultação níveis de detalhes | Não | Não | Sim |
| Vendedor | Compucenter | Intertec | Arthur Young |
| Fabricante | Yourdon, Inc. | FG & A | KnowledgeWare |
| Concepção BDD | MER | MER | MER com normalização para gerar estruturas de BDD Relacionais e Hierárquicas |
| Linguagem textual de especificação | meio-formal do Português Estruturado | meio-formal do Português Estruturado | Através do Action Diagrammer |
| Ajuda de contexto | só de operação | de operação | de operação e de conceitos dos métodos |

projeto REMORA (1980) até hoje. As publicações sobre CASEs foram de dois tipos: um é constituido por propostas e descrições de CASEs e o outro é constituido por analises gerais sobre CASEs. Outras publicações permitiram ter-se uma idéia do estagio atual das metodologias e da tecnologia de CASEs.

## 3. MACROMETODOLOGIA MIND-BFB:

"Ninguém pode ser convencido ou negado se não for pelos argumentos que ele admite." ABELARD

### 3.1. Apresentação da MACROMETODOLOGIA:

A MIND-BFB (Macrometodologia INtegrada de Desenvolvimento de sistemas do Banco Francês e Brasileiro) é uma metodologia que abrange as diferentes atividades de construção e manutenção de sistemas, num nível acima da metodologia REMORA, a qual utiliza como um de seus componentes [BAR1 87].

Ela nasceu com a constatação de que uma única metodologia não pode cobrir facilmente todos os aspectos de um problema informático.

Uma metodologia é caracterizada por quatro componentes integrados [ROL 83]:

- **Modelo:** fornece os conceitos com os quais é possível compreendermos um dado tipo de problema.

- **Linguagem:** fornece a sintaxe e a semântica que permitem expressar o problema em termos de suas características e restrições, com base nos conceitos do modelo.

- **Ferramentas:** são recursos técnicos que dispomos para a modelagem do problema e sua descrição através da linguagem.

- **Métodos:** são as etapas a serem seguidas e as ações a serem executadas para resolver o problema, usando os três primeiros componentes.

Então, se existisse uma metodologia universal, ela teria que possuir um modelo capaz de traduzir todos os problemas, uma linguagem que pudesse descrever todas as soluções possíveis, ferramentas que se adaptassem a qualquer ambiente e um método aplicável a diferentes situações e níveis de detalhes [BAR1 87].

Intuitivamente, isso parece impossível. Daí a proposta de macrometodologia dentro da qual as diferentes técnicas possíveis possam ser integradas e adaptadas aos recursos de cada instalação.

#### Componentes da macrometodologia:

Na vida de um SI três grandes problemas se colocam:

- **Controle de projeto:**
  planejamento e acompanhamento de recursos.

- **Especificação do problema:**
  descrição dos componentes e suas associações.

- **Construção da solução:**
  materialização em função dos recursos físicos existentes.

A **MIND-BFB** apóia-se, então, sobre três classes de metodologias que dão suporte para a solução de cada um desses problemas.

A seguir uma visão de cada uma dessas classes de metodologias que formam a macrometodologia.

**Metodologia de controle de projeto:**

Foi adotada a abordagem clássica das fases de atividades de realização de um projeto:

- ESTUDO PRELIMINAR (identificação do problema)

- ANTEPROJETO (definição do macro-contexto da solução)

- CONCEPÇÃO FUNCIONAL (descrição detalhada da funcionalidade da solução desejada)

- REALIZAÇÃO FÍSICA (programação)

- TESTES (qualidade da solução)

- IMPLEMENTAÇÃO (instalação no ambiente operacional)

- ACOMPANHAMENTO (revisão da solução em função das novas necessidades)

A inovação da **mind-bfb** é a divisão de cada uma dessas sete fases tradicionais em três níveis distintos, cada um preocupado com um conjunto de problemas diferentes. Os três níveis são: **o conceitual, o lógico e o físico.**

Assim, dado um nível, cada fase sucessiva detalha a anterior com o mesmo tipo de abordagem: sedimentação do processo de desenvolvimento de SI.

**Metodologia de especificação do problema:**

As bases para essa metodologia foram desenvolvidas no projeto REMORA. Existem algumas alterações da REMORA no contexto da MIND-BFB para adequação com a realidade do banco:

- Foi adotado a Terceira Forma Normal (3FN) no lugar da Terceira Forma Normal Permanente (3FNP) como forma de normalização dos objetos.

- Não faz parte do modelo a normalização dos eventos e das ações.

- Não é considerada a noção de classes para as três categorias de fenômenos (objetos, eventos e ações).

- Foi implementada outra linguagem de especificação (a MIND-BFB) diferente da ISDEL, já que têm objetivos diferentes. A MIND-BFB considera o uso da ferramenta e tem suas primitivas em português.

**Metodologia de construção da solução:**

Aqui são consideradas as técnicas informáticas em si. Essas técnicas são divididas em dois grandes grupos:

**- técnicas de uso geral:**
programação estruturada, engenharia de informação, sistemas especialistas, etc...

**- técnicas de uso local:**
restrições próprias à instalação: sistemas operacionais, compiladores, "hardware" (material), etc...

Dessas duas classes de técnicas, aquela que tende a evoluir mais rapidamente é a de uso geral, cuja evolução deve ser inserida na macrometodologia.

As técnicas dessa classe foram divididas em cinco grandes grupos:

- descrição de estruturas (Sub-Esquema estático conceitual)
- descrição de processos (Sub-Esquema dinâmico)
- construção de processos (técnicas de programação)
- utilização de gerenciadores de estruturas (Sub-Esquema estático físico)
- integração de processos e gerenciadores de estruturas (para levar em conta problemas de desempenho, sincronização e restrições físicas).

A integração das três metodologias (controle de projeto, especificação de problema e construção de solução) foi conseguida, para formar a macro, alocando para cada **par [nível,fase]** da metodologia de controle de projeto, as técnicas de construção cabíveis, apoiadas sempre nos conceitos da metodologia de especificação [BAR1 87].

Concluindo, a MIND-BFB surgiu de uma necessidade da própria industria. Ela está ainda ao nível das idéias e só foi detalhado o aspecto Especificação do Problema, para o qual foi proposto o modelo REMORA que é uma metodologia com fundamentos teóricos e práticos provados (ela se desenvolveu através de trabalhos conjuntos entre universidades e industrias francesas como THOMSON-CSF).

É proposta uma abordagem que envolve vários aspectos de várias metodologias, de forma que a macrometodologia seja aberta, dentro da qual as diferentes técnicas disponíveis possam ser integradas e adaptadas aos recursos disponíveis.

Depois desta breve apresentação da macrometodologia MIND-BFB, a seguir uma descrição suficientemente detalhada da metodologia REMORA para a qual é proposta uma ferramenta de apoio.

## 3.2. Introdução à metodologia REMORA:

### 3.2.1. Problemática da concepção dos Sistemas de Informação (SI):

Os SI são representações codificadas e parciais dos elementos constituintes da organização e das regras de gestão e funcionamento da organização. Uma tentativa de definição da semântica do Sistema de Informação levou [FOU 82] às seguintes definições:

Dados: são representações parciais dos constituintes da organização e da estrutura desses.

Programas: conjunto de instruções de dois tipos:
. instruções de E/S, que não têm conteúdo semântico em relação à organização e,
. instruções de tratamento que representam transformações reais que conheceram os constituintes e as regras de gestão da organização.

Comandos dinâmicos: a interpretação semântica é difícil. São comandos definidos a partir de considerações técnicas, econômicas ou de seguranca (como ou quando disparar um processo, por exemplo).

A qualidade dessas representações depende de suas naturezas em relação ao tempo (instantâneas ou históricas) e de suas dependências em relação às suas utilizações.

A representação fornecida por um SI é geralmente instantânea (o estado da organização em um dado instante).

Alguns SI fornecem versões sucessivas de dados, mas essas representações são armazenadas de um modo que não permite uma consulta direta. Elas se limitam aos dados e são parciais. Não existem representações históricas dos programas e dos comandos dinâmicos.

Os SI clássicos, organizados em termos de arquivos são, em geral, muito dependentes do uso imediato.

As Bases de Dados foram introduzidas nos SI para reduzir essa dependência entre dados e programas. A independência foi realizada pela introdução de vários níveis de representação. A busca de solução só foi feita ao nível dos dados e as Bases de Dados não trouxeram de fato nenhuma solução quanto à concepção e a representação dos programas e dos comandos [FOU 82].

Para melhorar as representações dos SI, é necessário:

— que o tempo seja considerado na representação dos comandos e dos programas. Assim, a representação histórica não vai só conter os estados sucessivos dos constituintes mas também os fatos reais que ocasionaram as mudanças de estado.

— que a representação dos comandos dinâmicos (regras de gestão e de funcionamento) seja feita, como é o caso nas Bases de Dados, a um nível abstrato, de maneira independente da utilização do SI.

A solução proposta para resolver esse problema foi a escolha do conceito de modelo para representar o SI.

O modelo conceitual é um conjunto de conceitos que permite expressar o SI [FOU 82]. Os componentes do modelo são os dados, os programas, os comandos e o tempo.

O modelo conceitual deve ser formal para assegurar a minimalidade da representação, verificar sua coerência e sua completeza. Ele deve ser também "orientado à concepção", isto é, os conceitos do modelo devem ajudar o projetista a extrair da análise da realidade, ou do enunciado do problema, o núcleo minimo que o caracteriza. O Esquema Conceitual (EC) é a expressão abstrata do SI [THI 85], resultado da modelagem conceitual. Ele tem papel importante tanto durante a concepção quanto durante o uso do SI.

Durante a concepção, o EC serve de ponto de partida para a busca de uma solução técnica de implementação do SI num ambiente técnico dado.

Durante a utilização e a evolução do SI, o EC permite uma interpretação semântica dos resultados, é reutilizável como ponto de partida para a escolha de uma nova solução técnica e permite determinar facilmente os efeitos da evolução real da organização sobre o SI. Essa preocupação já foi considerada na área das Bases de Dados:

As Bases de Dados de primeira geração possuiam duas características: não distinguiam os níveis e o algoritmo refletia a representação.

A primeira caracteristica é que não existem níveis diferentes de modelagem (conceitual, lógico ou físico). A representação da informação é ligada à sua representação física. O que significa

que se modificarmos a representação interna dos dados por qualquer razão, temos que modificar os programas que tem acesso a essas representações.

A segunda caracteristica é que nos programas que manipulam esses dados, os algoritmos descrevem a maneira de percorrer a Base de Dados, o que dificulta e a programação e as consultas.

Foi no início dos anos 70 que começaram os primeiros trabalhos de pesquisas e de realização de protótipos de Sistemas Gerenciadores de Bases de Dados (SGBD) de tipo relacional [DEM 85]. O objetivo essencial destes gerenciadores de segunda geração é oferecer um nível conceitual nitidamente distinto do nível interno.As informações são representadas com a ajuda de noções simples da álgebra: relações, conjuntos, operadores algébricos sobre conjuntos ou operadores lógicos e quantificadores. Essa abordagem possui duas características interessantes:

a) seu carácter conceitual:
Que permite construir uma representação das estruturas de dados independentes de qualquer consideração relativa à implementação física.

b) seu carácter formal:
Que conduz a construir esquemas corretos para os quais será possível verificar se possuem qualidades formais que conduzem à não redundância, à fidelidade e à completeza.

O modelo relacional foi então escolhido pelos autores de REMORA em razão dessas qualidades para representar os elementos do modelo que vamos ver a seguir. Existem duas fases de modelagem: a Modelagem Descritiva seguida da Modelagem Conceitual, ambas incluidas na fase de criação do Esquema Conceitual. Em seguida será vista a segunda fase que corresponde à definição do Nível Interno do sistema.

### 3.2.2. O Modelo Descritivo:

O modelo descritivo serve para fazer a descrição da organização e, depois, para sua análise.

A organização é um sistema real organizado para satisfazer um objetivo. Ela é composta de dois sub-sistemas:

sub-sistema causal, composto de unidades causais, como processos tecnológicos, por exemplo.

sub-sistema dos componentes, composto de unidades atrás de objetivos (ex: serviços, homens), definindo e usando o primeiro [FOU 84].

Aqui, só interessa o sub-sistema causal.

Interrogação sobre as categorias de fenômenos pertinentes para descrever um sistema dinâmico leva a responder às três perguntas seguintes:

-Por que é definido o estado da organização em um dado instante?

**objeto:** componente concreto ou abstrato da organização que possa ser particularizado. Exemplo: conta.

- O que modifica o estado da organização a um dado instante?

**operação:** Ação que pode ser executada em um dado momento na organização e que modifica o estado de um ou vários objetos. Exemplo: creditar-conta.

- Quais são as origens dessas modificações?

**evento:** Qualquer modificação que acontece em um dado momento. É a constatação de uma mudança de estado de um ou vários objetos, consequência da execução de alguma operação. Exemplo: depósito.

Assim, numa perspectiva dinâmica, temos que representar três categorias de fenômenos, descritos através suas propriedades, como objetos, eventos e operações (aqui é utilizado como sinônimo de ações).

- Para completar a dimensão dinâmica, são propostas três categorias de associações, entre as três categorias de fenômenos:

constata (objeto,evento): indica qual evento decorre de uma mudança de estado de qual objeto.

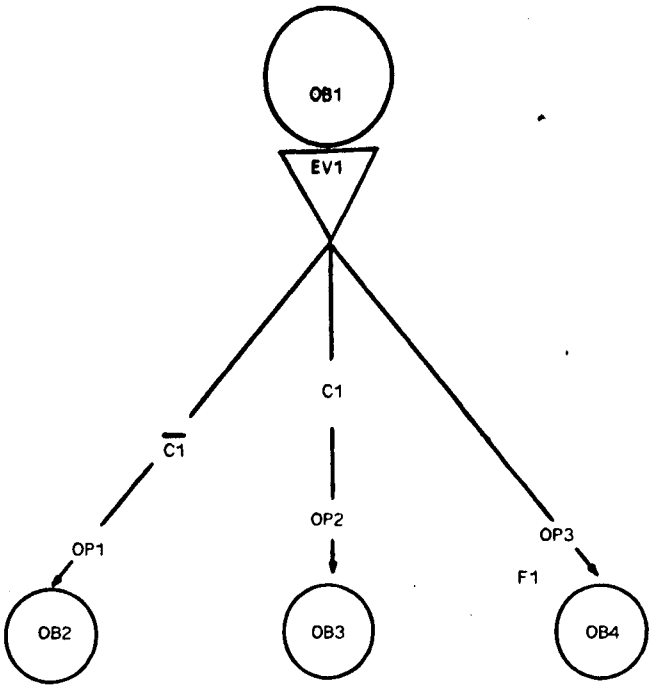
dispara (evento, operação): indica qual evento dispara a execução de qual operação.

modifica (operação,objeto): indica qual objeto é modificado por qual operação.

As três categorias de fenômenos que são a base do modelo são representadas graficamente da seguinte maneira:

objeto: circulo
evento: triângulo
operaçao: flecha

**A figura 4a** ilustra esses conceitos e a **figura 4b**, a seguinte definição causal da organização:



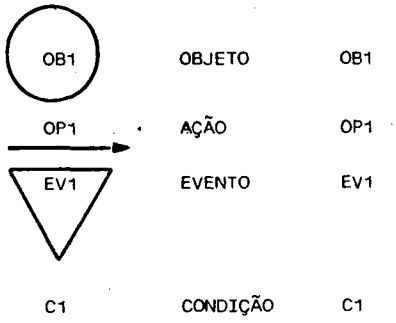




**Figura 4a:** *Exemplo de ciclo dinâmico ilustrando os elementos usados na meodelagem conceitual da metodologia REMORA.*

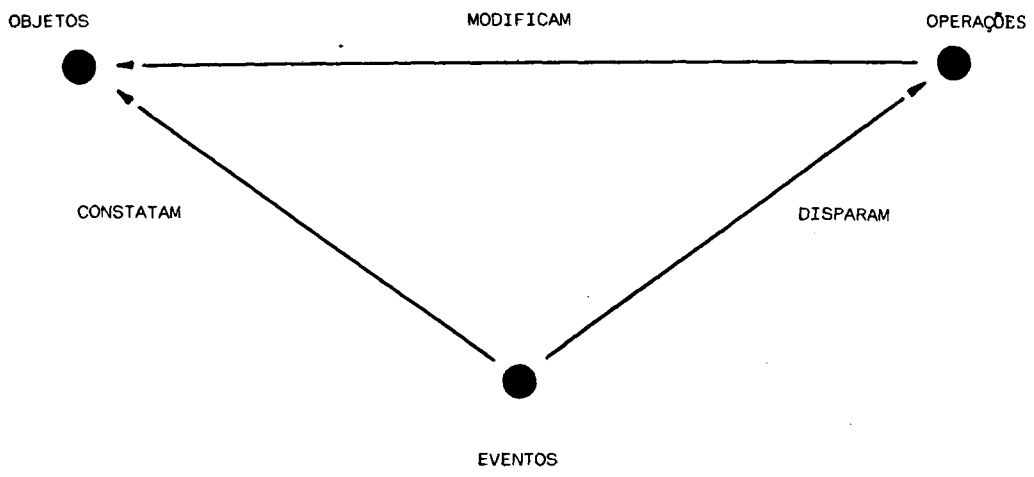



Figura 4b: *Ilustração da definição causal da organização.*

**Os eventos causam o disparo das operações que modificam os objetos. Essas modificações poderão ser consideradas como novos eventos.**

A modelagem descritiva consiste então em descrever o sistema através dos três elementos (objeto, operação e evento) e suas três associações (constata, dispara e modifica).

### 3.2.3 O modelo conceitual:

A representação resultante do modelo descritivo apresenta certamente incompleteza, incoerência e redundância. O modelo conceitual deve ajudar na maneira de representar o SI, a fim de chegar a um esquema conceitual canônico [FOU 82]. Isto é, coerente, não redundante, fiel, completo e simples.

O modelo conceitual deve satisfazer, então, duas necessidades: ser formal e poder ser representado fácil e harmoniosamente. Para isso,

- foi escolhido o modelo relacional
- foi introduzida a noção de tipo para diferenciar as categorias
- foi introduzido o tempo

Os três tipos de relações, **c-objeto c-operações e c-eventos** (c-para conceitual), são expressas sob uma forma normalizada especial: a Forma Normal Permanente (FNP) que é definida abaixo. Antes, são definidos alguns conceitos básicos.

São apresentados aqui os principais resultados explorados do modelo relacional na modelagem conceitual da REMORA.

O modelo relacional conduz a uma representação das informações sob forma de tabelas de valores chamadas de relações. A relação é o conceito de base do modelo. Os outros conceitos têm por objetivo definir propriedades das relações a fim de chegar a uma melhor representação.

Sejam n conjuntos D1,D2,...,Dn. Uma relação R sobre estes n conjuntos é um sub-conjunto do produto cartesiano D1xD2x...xDn. Os Di são chamados de domínios da relação. Para diferencia-los, a cada um deles é associado um nome diferente, estes nomes são os atributos da relação. Um esquema de relação define um tipo de relação por:

. um nome de relação,
. n nomes diferentes de atributos A1,A2,...,An,
. n conjuntos D1,D2,...,Dn, domínios de Ai,
. n conjuntos de restrições de integridade.

Uma restrição de integridade é uma propriedade que deve ser verificada por toda relação pertencente a um tipo definido pelo esquema. As restrições de integridade podem ser formalmente definidas por:

. a definição de uma chave,
. dependências entre atributos,
. predicados sobre domínios.

Segue a definição de noções de projeção e super-chave, necessárias para definir a noção de chave. Sejam:

- uma relação de esquema R(A1,...,An),
- X um sub-conjunto de dos atributos: X = [Ai1,...,Aip] de R,
- uma n-upla de R <a1,...,an>.

A projeção de R segundo X da n-upla é a p-upla <ai1,...aip>.

Um conjunto de atributos de X é uma super-chave de uma relação R se duas n-uplas diferentes de R têm projeções diferentes sobre X. X é uma chave se ela é uma super-chave mínima, isto é, não existe nenhum conjunto Y incluido em X que seja ele mesmo super-chave.

Sejam X e Y dois sub-conjuntos disjuntos de A, onde A e o conjunto dos atributos de R. Existe uma dependência funcional (DF) entre X e Y em R, notada X ---> Y, se e somente se quando duas n-uplas têm mesma projeção sobre X, eles têm igualmente mesma projeção sobre Y. Uma DF X ---> Y é elementar se não existe X' incluído em A tal que X ---> Z existe, Z ---> Y existe e Z ---> X não existe em R. Uma DF X ---> Y é canônica se Y é um conjunto de atributos contendo um único elemento [ROL 88].

A seguir a definição da Terceira Forma Normal que será retida como forma final das relações na metodologia MIND-BFB. Antes desta definição, serão vistas a Primeira e a Segunda Formas Normais.

Uma relação está na Primeira Forma Normal (1FN) se todos os seus atributos não pertencentes à chave estão em DF com esta.

Uma relação está na Segunda Forma Normal (2FN) se ela está na 1FN e se seus atributos não pertencentes à chave estão em DF elementar com esta.

Uma relação está na Terceira Forma Normal (3FN) se ela está na 2FN e se seus atributos não pertencentes à chave estão em DF elementar e direta com esta.

A 3FN de Codd [COD 70] possui dois grandes interesses praticos que são a facilidade de atualização e sobretudo a eliminação de redundancias na representação.

Além da definição das formas normais, existem diversos métodos de normalização das relações. Um deles é o de Bernstein [FOU 82] que é apresentado a seguir:

O método permite passar de um esquema inicial composto de um único esquema de relação a um esquema conceitual composto de n esquemas de relação na 3FN.

Seja So = [Ro(A),A,Fo] o esquema conceitual inicial, com A: conjunto dos atributos de Ro, D: conjunto dos domínios dos elementos de Ai. O algoritmo de Bernstein permite passar de So a um Sf:

Sf = [ <Ri(Ai),Di,Fi>, i=1,n ], com Ai: conjunto dos atributos de Ri, Di: conjunto dos domínios de Ri e Fi: conjunto das DF existentes entre os elementos de Ai.

O algoritmo e composto de quatro passos:

1) A busca de uma cobertura mínima, Fm, do grafo associado a Fo.

2) A partição de Fm em sub-conjuntos Fi, i=1,n formados das DF com o mesmo lado esquerdo.

3) A fusão das chaves equivalentes e a eliminação das dependências transitivas resultantes que conduzem a um conjunto de grupos F'm, i=1,n.

4) A construção das relações Ri de Sf, uma para cada F'i. O lado esquerdo do grupo F'i é a chave da relação assim gerada.

Dois resultados de DELOBEL [DEL 73] enunciados em [FOU 82] simplificam o algoritmo de BERNSTEIN visto. O primeiro é que o conjunto de DF elementares, diretas e canônicas da relação Ro constitue uma cobertura mínima. Com esse resultado, o algoritmo se reduz ao passo 2). O segundo é que se Fm é não tem circuito, a solução obtida é única.

As vantagens do método são:

- O uso do conceito único de DF, simples de entender e que de outro lado obriga o projetista a bem entender o que está concebendo.

- Suporte gráfico (que será visto no exemplo a seguir) facilitando o uso manual nos casos simples.

- Possibilidade de automação nos casos volumosos e complexos

- Unicidade da solução obtida.

Por exemplo, seja Ro = pedido (NOPED, NOCLI, ENDERCLI, NOPROD, PRECO, QDE, DATAPED), e as retrições

NOPED ---> NOCLI

NOCLI ---> ENDERCLI

NOPROD ----> PRECO

NOPED ----> DATAPED

NOPED, NOPROD ----> QDE, constituindo Fo.

Pode-se observar que Fo só contém DF elementares, diretas e canônicas.

O grafo associado a Fo encontra-se na **figura 5.**

Essa partição de Fo em sub-conjuntos Fi, i=1,n formados das DF tendo mesmo lado esquerdo conduz aos sub-grafos inclusos nos retangulos desenhados no grafo inicial. Se deduz então o esquema Sf formado pela coleção de relações seguintes:

R(NOPED, DATAPED, NOCLI)

R(NOCLI, ENDERCLI)

R(NOPROD, PRECO)

R(NOPED, NOPROD, QDE)

Os autores da REMORA fizeram uma extensão do Modelo Relacional visto, introduzindo o tempo. O uso, hoje, do Modelo Relacional conduz a uma representação instantânea da realidade. A evolução dos fenômenos no descurso do tempo implica adições, modificações e supressões de n-uplas nas relações. A Base de Dados (BDD) não conserva traços dessa evolução [ROL 88].

É proposto aqui um meio rigoroso para a introdução dessa dimensão da realidade, permitindo uma representação não redundante, minima e fiel. De um lado, é preciso representar as propriedades temporais dos fenômenos reais por constituintes da relação (data de mudança de estado, data de criação, data de supressão, etc...). De outro lado, conservar as versões sucessivas das relações no descurso do tempo. Isso possibilitará a consulta dos estados sucessivos deste fenomeno. Para chegar a essa representação e o SI ainda satisfazer ao critério de minimalidade necessario para o SI, foram introduzidos dois conceitos:

- Relação Dependente do Tempo e

- Relação Permanente.

Seja um esquema de relação S(A, D, F) com A: conjunto dos atributos, D: conjunto dos domínios e F: conjunto das dependências funcionais. Seja um conjunto T munido de uma relação de ordem cujos elementos representam datas. Uma relação dependente do tempo é uma função R de T no conjunto das partes

D1 x D2 x ... x Dt x ... x Dn tal que: V t de T, R(t) pertence a S.

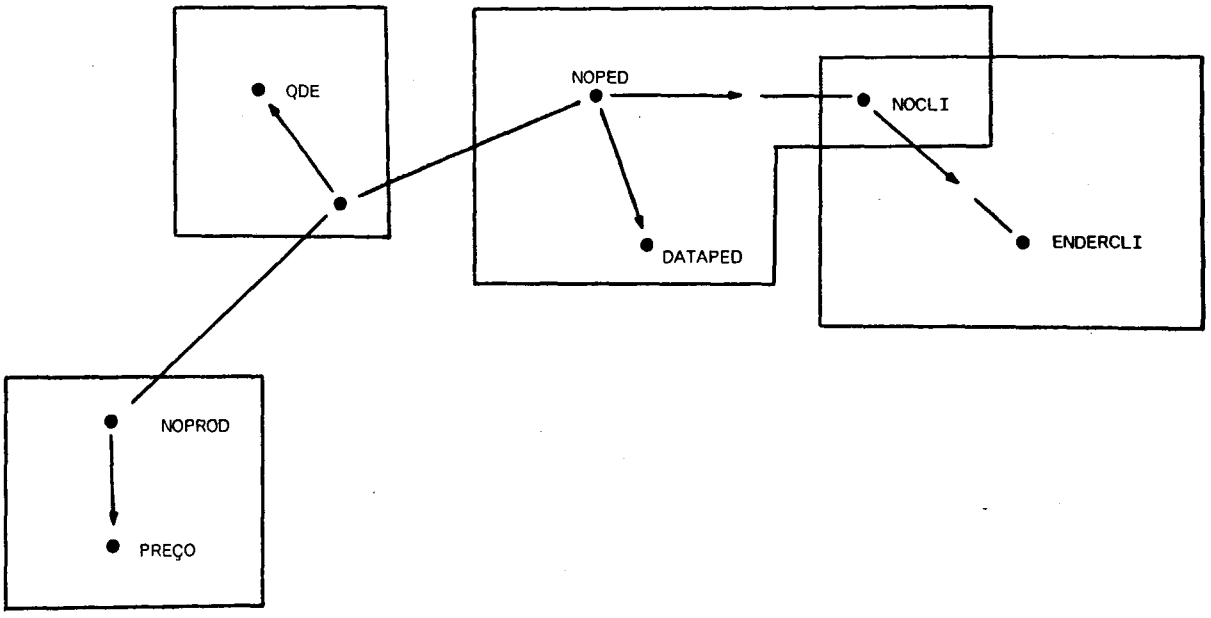



Figura 5: *O grafo das Dependências Funcionais associado à relação Pedido.*

Uma relação permanente e uma relação dependente do tempo tal que:

V (t,t') de T, t' < t ==> R(t') e incluido em R(t). Isto é, numa relação permanente, a única modificação que pode aparecer entre duas versões sucessivas da relação é a adição de n-uplas.

Assim foi conseguido a representação das propriedades temporais, a conservação das versões sucessivas e restá a satisfazer o critério de minimalidade da representação. Para isto, foram definidos dois conceitos: a Dependência Funcional Permanente (DFP) e a Terceira Forma Normal Permanente (3FNP).

Sejam: R(A) dependente do tempo,
X e Y dois sub-conjuntos de A,
t de T,
Rt, a extensão de R ao instante t,
existe uma DF entre X e Y em R se V t de T, existe em Rt uma DF entre X e Y.

Um conjunto RT é assim definido:

RT = União dos Rt[X,Y]: {<x,y>: existe t: <x,y> pertence a Rt[X,Y]}, t pertence a T, com Rt[X,Y] = {<x,y>: <x,y> pertence a Rt}.

Uma DF entre X e Y é uma Dependência Funcional Permanente (DFP) se existe em RT uma DF entre X e Y.

A existência de uma DFP entre dois constituintes X e Y de uma relação dependente do tempo significa que no descurso do tempo, a um mesmo valor de x ( do domínio de X) é sempre associado o mesmo valor de y (do domínio de Y). Se X e Y representam duas propriedades de um mesmo fenômeno real, a DFP significa que essas duas propriedades são sempre valorizadas ou modificadas simultaneamente: elas têm o mesmo comportamento dinâmico.

É possivel definir Formas Normais para as relações permanentes baseando-se na noção de DFP e, em particular, a Terceira Forma Normal Permanente (3FNP). A relação permanente na 3FNP é a forma escolhida para o Modelo Conceitual no projeto REMORA.

Por exemplo, seja a relação CLIENTES de esquema

CLIENTES (NOCLI, ENDERECO, NOTEL, NOMECLI)

agrupando numero, endereco, numero de telefone e nome dos clientes. Tem-se as seguintes DF:

NOCLI ---> ENDERECO

NOCLI ---> NOMECLI

NOCLI ---> NOTEL

Sejam as extensões dessa relação aos instantes t1, t2 e t3 na **figura 6.**

É imediato verificar sobre essas três extensões que tem-se para as três relações clientes t1, clientes t2 e clientes t3 as DF citadas acima.

Dessas três DF, só a DF NOCLI ---> NOMECLI é permanente para a relação CLIENTES.

Portanto, é possível agora definir precisamente os conceitos de c-objeto, c-operação e c-evento:

C-objeto é uma relação de tipo permanente. Uma relação permanente é uma relação na Terceira Forma Normal onde cada atributo está em Dependência Permanente com a chave da relação.

C-operação é uma relação permanente obedecendo às seguintes restrições:

c-operação --> c-objeto

c-operação --> tipo-mudança

c-operação --> texto-operação, usando a notação da Dependência Funcional.

Isso significa que uma c-operação modifica um único objeto e de uma única maneira. Também que uma c-operação representa uma regra de gestão da organização.

C-evento é uma relação permanente satisfazendo as seguintes retrições:

c-evento --> c-objeto

c-evento --> tipo-mudança

c-evento --> predicado

c-evento -->> c-operação, usando a notação da dependência multivalorada.

Isto é, um c-evento se refere a um c-objeto, corresponde a uma única mudança de estado, a um único predicado e gera uma ou várias operações.

O esquema conceitual resultante é uma coleção de relações pertencentes às três categorias de fenômenos: **c-objeto, c-operação e c-evento** e suas associações.

Clientes T1

| NOCLI | ENDEREÇO | NOTEL | NOMECLI |
|---|---|---|---|
| 1 | Rio | 712 | José |
| 2 | Belém | 832 | João |
| 3 | Natal | 764 | Mário |

Clientes T2

| NOCLI | ENDEREÇO | NOTEL | NOMECLI |
|---|---|---|---|
| 1 | Poços | 243 | José |
| 2 | Belém | 832 | João |
| 3 | Recife | 195 | Mário |

Clientes T3

| NOCLI | ENDEREÇO | NOTEL | NOMECLI |
|---|---|---|---|
| 1 | Poços | 325 | José |
| 2 | Belém | 832 | João |
| 3 | Natal | 241 | José |
| 4 | Natal | 842 | Carlos |

Figura 6: *Exemplos de extensões da relação Clientes em três instantes diferentes.*

Essa coleção de relações é completada por um conjunto de restrições de integridade. A coleção pode ser subdividida em dois subesquemas.

-coleção dos c-objetos (**Sub-Esquema estático**): corresponde ao **esquema dos dados.** Constitui a visão estática da organização através de suas entidades e dos atributos dessas entidades.

-coleção de c-operações e c-eventos (**Sub-Esquema dinâmico**): corresponde à visão dinâmica da organização. Ela representa as regras de evolução da organização através das interações entre as três categorias.

Poucas abordagens definem o SI sob um escopo dinâmico como REMORA [ROL 82].

Na área de gerenciamento de dados, os SGBD introduziram uma grande mudança nas tarefas dos projetistas.

Os SGBD são ferramentas que criam e gerenciam Bases de Dados a partir de uma especificação apropriada da estrutura dos dados fornecida pelo projetista. Para alguns SGBDs, a descrição da Base de Dados se limita à dimensão semântica. A tarefa do projetista é dar uma representação completa da realidade sem nenhuma referência à implementação física mas, por outro lado, a tarefa do SGBD é muito complexa.

Quando se descreve a estrutura de dados, o SGBD escolhe automaticamente todos os programas de controle e de manipulação (por interpretação da descrição), o que minimiza os erros nesses programas. Na hora da utilização, os usuários finais se comunicam com a Base de Dados através de linguagens não algorítmicas.

Os autores da REMORA acham necessário proceder a uma abordagem idêntica em relação à MSI (Máquina de Sistema de Informação) que é uma máquina abstrata em projeto para assistir o projetista utilisando a REMORA [FOU 82]. O SI definido segundo essa metodologia é uma representação mais completa da realidade do que a Base de Dados pois inclui tanto a dimensão estática (que corresponde à Base de Dados) quanto a dinâmica. Essa completeza da representação a ser feita implica numa complexidade da MSI. A MSI deve assegurar várias funções do SI:

- Geração da estrutura interna
- Criação
- Gerenciamento da evolução
- Utilização

Os autores de REMORA admitem a hipótese de que é possível assistir todo o ciclo de vida do SI através de um "CASE".

Esse sistema combinaria as grandes capacidades heurísticas do homem com aquelas da máquina, de gerar soluções para problemas cujas complexidades foram vencidas.

Da MSI faz parte o autômato da dinâmica [ROL 82] da realidade. É uma máquina abstrata que toma conhecimento das modificações ocorridas no SI e reage em função delas:

- Identifica os eventos produzidos no SI
- Determina as operações que deverão ser disparadas
- Dispara essas operações e controla suas execuções corretas
- Verifica se as mudanças de estado são eventos
- Atualiza o SI (ver figura 7).

Foi associada à metodologia REMORA uma linguagem de descrição. O objetivo da linguagem ISDEL (Information System DEsign Language) é:

1) Permitir a descrição do EC (as relações e os controles de integridade associados).
2) Servir para a Máquina de Sistema de Informação (MSI).

Para isso,

1) Foi escolhida uma linguagem relacional completa que permite fazer a descrição e a manipulação da Base De Dados.
2) ISDEL contém construções derivadas de linguagens de programação de terceira geração (PASCAL, PASCAL-R, etc.).

Com essas construções, podemos definir os textos das c-operações e das condições de aceitação dos eventos e de disparo das operações [LEI 80] [ROL 82] [ROL 88].

### 3.2.4. O MODELO INTERNO:

Depois da descrição conceitual, poderá vir a necessidade de se fazer o Esquema Interno, considerando:

- A utilização específica dos usuários
- Os parâmetros técnicos.

O Esquema Interno é composto por dois Sub-Esquemas:

- O Sub-Esquema de Estruturação de Dados
- O Sub-Esquema de Sincronização de Processos (ver figura 7)

A modelagem do Sub-Esquema de Estruturação de Dados considera:

. O Sub-Esquema Estático Conceitual
. As necessidades permanentes e aleatórias dos usuários
. As capacidades e limites do SGBD que vai operar.

Surgem aqui duas dificuldades que são o levantamento das necessidades e como escolher dentro do grande número de soluções possíveis. A modelagem passa a ser um compromisso crítico entre a redundância e a complexidade das estruturas que leva a um esforço

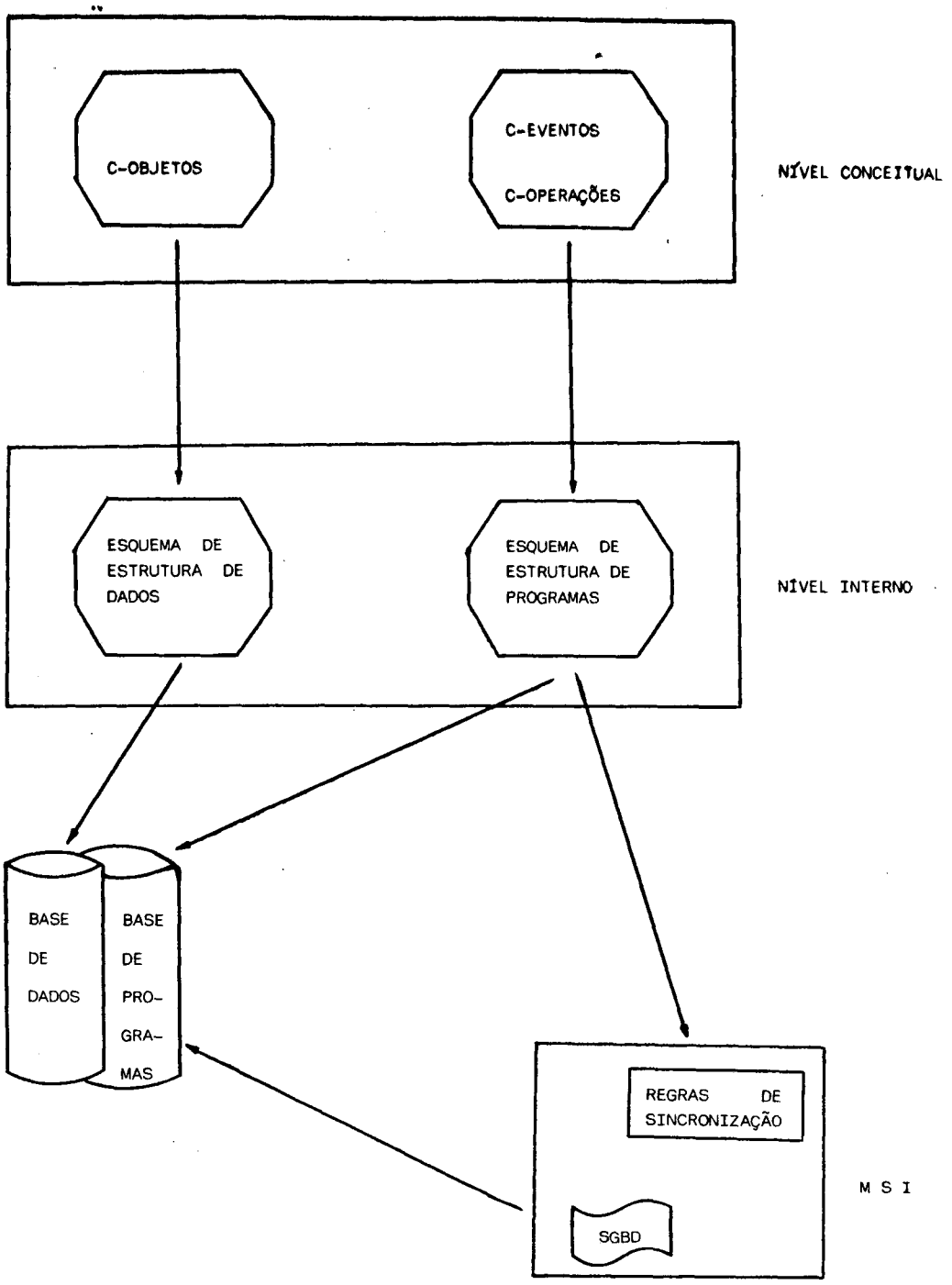



Figura 7: *Derivação do Modelo Interno a partir do Modelo Conceitual.*

suplementar em programação. É recomendado (para conseguir maior portabilidade) introduzir as características específicas do SGBD o mais tarde possível.

Para isso existirá um nível lógico, que vai permitir construir um esquema padrão contendo todos os modelos de dados existentes, e um nível físico para converter o esquema lógico num físico considerando as especifidades do SGBD a ser utilizado. BARALDI definiu um modelo de conversão do nível conceitual para o nível físico correspondente ao SGBD ADABAS [BAR2 87].

A modelagem do Sub-Esquema de Sincronização de Processos considera:

- O Sub-Esquema Dinâmico Conceitual
- As condições de uso definidas pelos usuários
- O ambiente técnico de operação do SI

Como foi o caso do esquema estático, aqui também o projetista enfrenta algumas dificuldades em termos de compromissos entre recursos técnicos disponíveis, ideal do modelo e esforço suplementar de programação.

É proposto um Modelo de Sincronização de Processos lógicos e um conjunto de regras para converter o Sub-Esquema Dinâmico Conceitual no lógico, chamado Esquema de Sincronização de Módulos (ESM). O modelo é baseado em dois conceitos que são:

- <u>Módulo</u>: sequência de instruções executadas numa certa ordem e tendo uma única entrada.

```
 ---------
| MOD 1 |
 ---------
```

- <u>Disparo</u>: condição para execução de um e só um módulo.

```
  /\
 /  \---->
 \D1/
  \/
```

As duas restrições sobre os dois conceitos do modelo implicam que qualquer Esquema de Sincronização de Módulos é constituído a partir de três estruturas básicas que são "linear", "garfo" e "join" **(ver figura 8)**

Existem cinco regras para converter o esquema dinâmico conceitual no ESM.

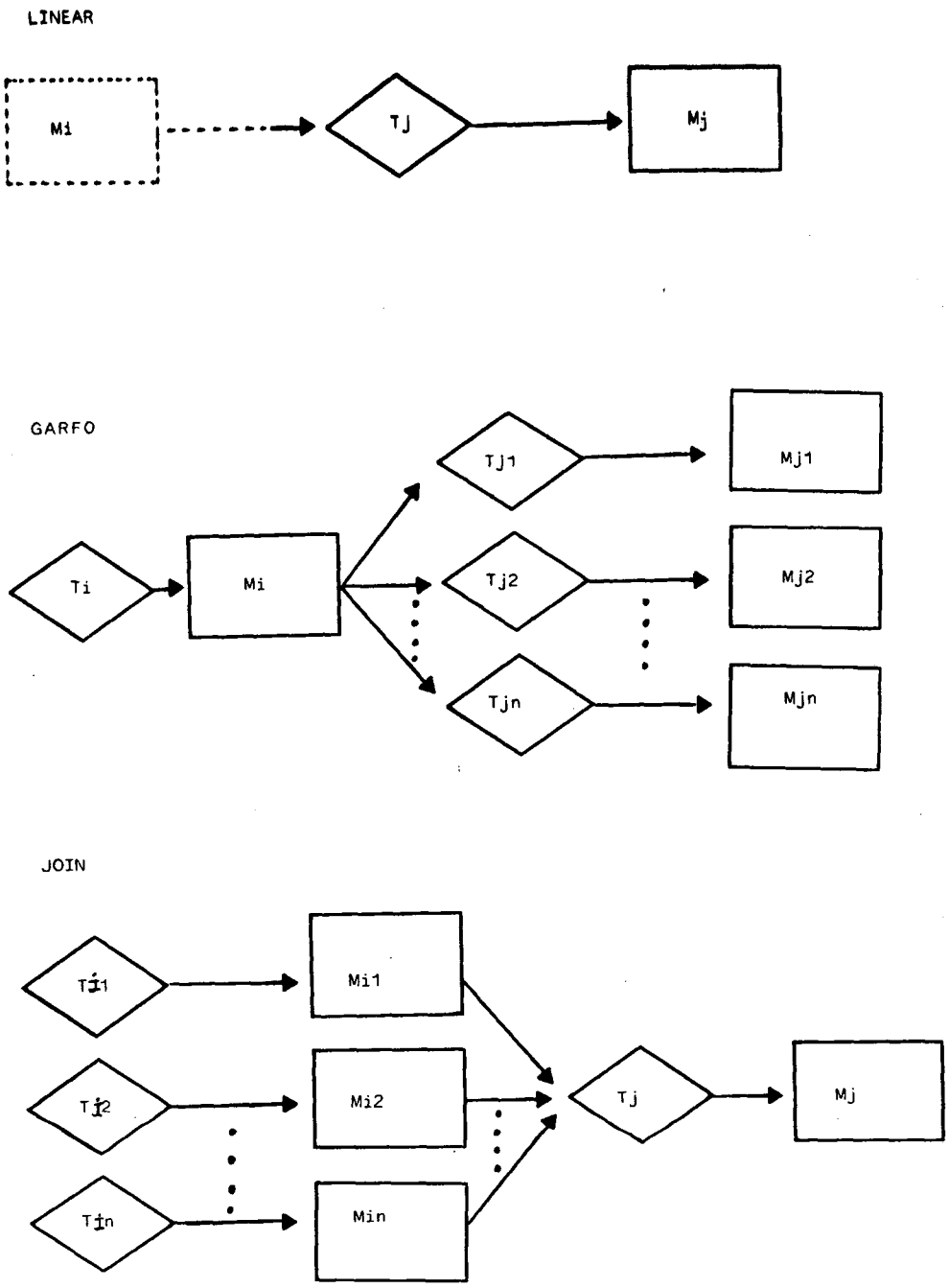



**Figura 8:** *As três Estruturas Básicas do Esquema de Sincronização de Módulos.*

regra_1: determinação_dos_módulos_elementares:

Um ciclo dinâmico é composto de:

- Um evento EVj
- Um conjunto de operações (OPj)k, que EVj dispara
- Os c_objetos modificados pelas operações (OPj)k

Um módulo elementar Mj é o conjunto de instruções implementando as operações (OPj)k. Seu parâmetro de entrada é o c-objeto OBj constatado por EVj e seus parâmetros de saída são os c-objetos (OBj)k modificados pelas operações (OPj)k fontes de outros módulos (ver figura 9).

regra_2: concatenação_dos_módulos_elementares:

São definidos aqui dois tipos de dependência entre eventos: vamos considerar um ciclo dinâmico

```
            opj
EVi ------------ > (obj)EVj
```

. DCDS (Dependência Cronológica Direta Sistemática):

EVi está em DCDS com EVj ( denotada com EVi --s--> EVj ) se incondicionalmente, a ocorrência de EVi resulta na ocorrência de EVj.

. DCDC (Dependência Cronológica Direta Condicional):

EVi está em DCDC com EVj ( notada EVi --c--> EVj ) se a ocorrência de EVj não é sistemática mas uma eventualidade que depende das condicões de disparo das operações OPi ou das condições de aceitação de EVj.

Depois de montar o grafo das dependências cronológicas de um dado esquema conceitual dinâmico, aplicamos a seguinte regra:

1) EVi ---s---> EVj ==> Mi.Mj (concatenação de Mi e Mj)

2) EVi ---c---> EVj ==> escolher entre

Mi.Tj.Mj = Mi.Mj e
Mi e Mj separados.

regra_3: Definição_dos_disparos:

O disparo Tj é a condição associada à execução do módulo Mj, resultando da aplicação das duas primeiras regras. Essa condição é uma transformação do c-evento que gerou Mj. Isso significa que um disparo é associado a só um módulo (ver figura 8).

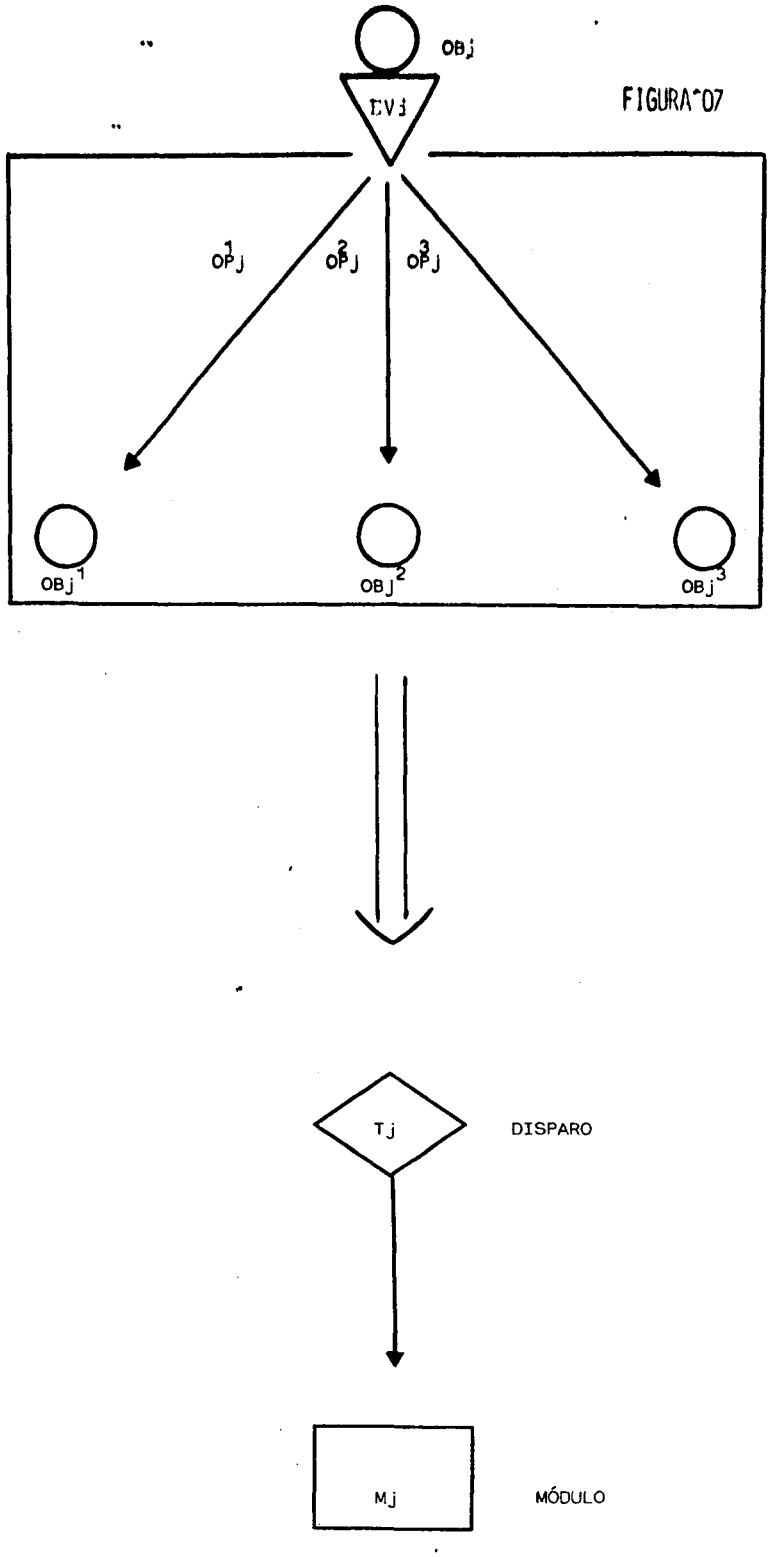



**Figura 9:** *Determinação de Módulos Elementares.*

regra 4: Definição de módulos independentes:

Se o ambiente técnico não facilitar mecanismos de sincronização, pode-se definir módulos independentes. Isto é, construir um módulo para cada componente conexo dos grafos das dependências cronológicas (condicionais ou sistemáticas).

regra 5: Escolha de eventos:

A escolha pelos usuários finais da validação dos eventos em tempo real ou em lotes poderá modificar o esquema de sincronização.

No caso do processamento por lotes, poderão ser introduzidos um módulo para armazenar as ocorrências de disparo, do disparo inicial, e um disparo condicionado à data do calendário.

### 3.2.5. UM EXEMPLO: CASO DE RESERVA EM HOTEL:

Inspirando-se no exemplo dado em [THI 85], é dado a seguir a definição de um problema (simples por razões didáticas). Seguem depois os passos (resumidos aqui e detalhados no item 4.5.) da sua resolução manual.

Se trata da concepção de um sistema para automatizar a reserva de quartos num hotel. Para isso, será necessário gerenciar os pedidos de reservas feitos, os dados dos clientes, as disponibilidades dos quartos e os eventuais cancelamentos que poderão occorer. Por razões de simplicidade, não serão assumidas as listas de espera. Se o cliente for novo, o sistema deve arquivar seus dados pessoais. Novos quartos entregues, desocupação de um quarto por um cliente e o cancelamento de um pedido deverão atualizar as disponibilidades.

Definição do Sub-Esquema Estático:

Está suposto que o projetista conseguiu, sem detalhar aqui como, a partir de entrevistas, de análise de documentos e de estrutura de fluxos de informação, dispor de um inventário de objetos e suas propriedades que lhe pareceu caracterisar a aplicação.

QUARTO: Número Quarto,
Tipo Quarto,
Preço Quarto,
Estado Quarto,
Data Mudança Estado Quarto.

CLIENTE: Número Cliente,
Nome Cliente,
Endereço Cliente,
Data Início Reserva,
Data Fim Reserva,
Tipo Reserva.

RESERVA: Número Quarto,
Número Cliente,
Início Reserva,
Duração Reserva,
Custo,
Estado Reserva,
Data Mudança Estado Reserva.

O método de concepção que permite obter o EC é baseado no modelo relacional de dados e na transformação formal das relações, usando a propriedade matemática de Dependência Funcional. Essa transformção formal chamada de normalização de relações permite obter relações normalizadas. Relações normalizadas não têm nenhuma redundância na representação do problema e permitem verificar que a representação é coerente.

A seguir as fases principais do método de concepção:

1) representação relacional do problema:

descrever os objetos por relações em forma qualquer considerando que cada objeto transforma-se numa relação e cada propriedade,num atributo.

QUARTO (noq, tipoq, preçoq, estadoq, datamudq)

CLIENTE (nocli, nomecli, endercli, datainício, datafim, tipor)

RESERVA (noq, nocli, inícior, duração, custo, estador, datamudr)

2) construção do grafo das Dependências Funcionais:

O processo de normalização que permite transformar as relações obtidas no primeiro passo numa forma otimizada é baseado na existência de dependências funcionais elementares e diretas entre constituintes (ou atributos).

O grafo das dependências funcionais que explicita essas dependências está na **figura 10.**

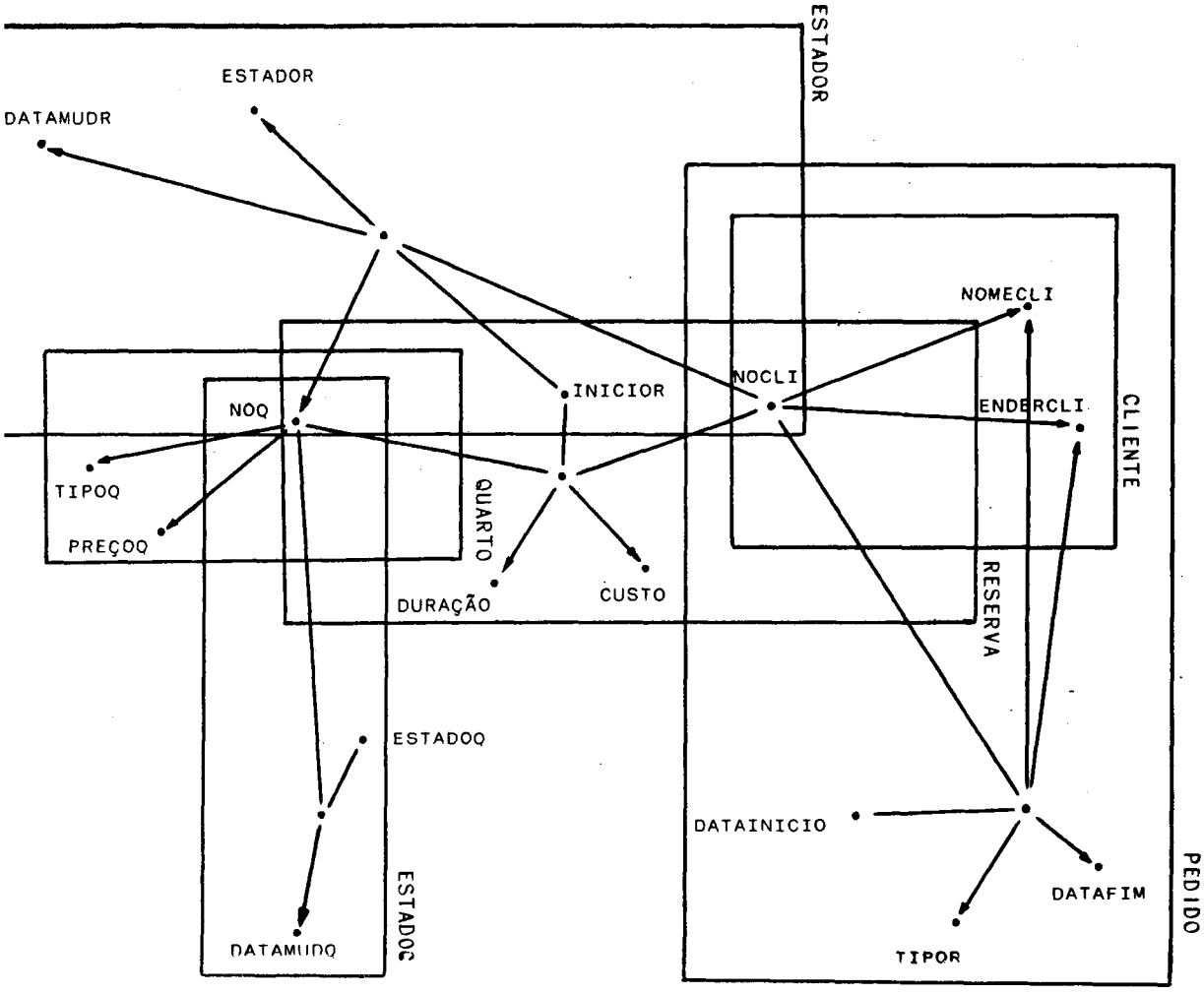



**FIGURA 10:** *O grafo das Dependências Funcionais associado ao exemplo: "caso de reserva em hotel".*

3) criando relações em forma normalizada:

A exploração do grafo das dependências funcionais permite criar uma relação para cada atributo (ou conjunto de atributos) origem de dependência funcional. Nessa relação, os atributos destinos de dependência funcional formam o resto dos atributos da relação. Os atributos origens (sublinhados nas relações) formam a chave.

O projetista passa então a dar um nome a cada relação assim obtida, pois a normalização é um processo de decomposição que faz aparecer objetos que eram até então ignorados pelos usuários e pelo projetista.

QUARTO (noq, tipoq, preçoq)

ESTADOQ (noq, estadoq, datamudq)

CLIENTE (nocli, nomecli, endercli)

PEDIDO (nocli, datainício, tipor, datafim, nomecli, endercli)

RESERVAS (noq, nocli, inícior, duração, custo)

ESTADOR (nocli, inícior, noq, datamudr, estador)

Além destes objetos, é necessário considerar um objeto fictício, calendário, cujas mudanças de estado particulares poderão originar alguns eventos no modelo:

CALENDARIO (dia).

Uma análise dos domínios dos atributos leva a definir os seguintes tipos:

Tipoq: "simples" / "luxo" / "médio"

Tipor: "simples" / "luxo" / "médio" / "cancel"

Nome: 30 caracteres alfabeticos

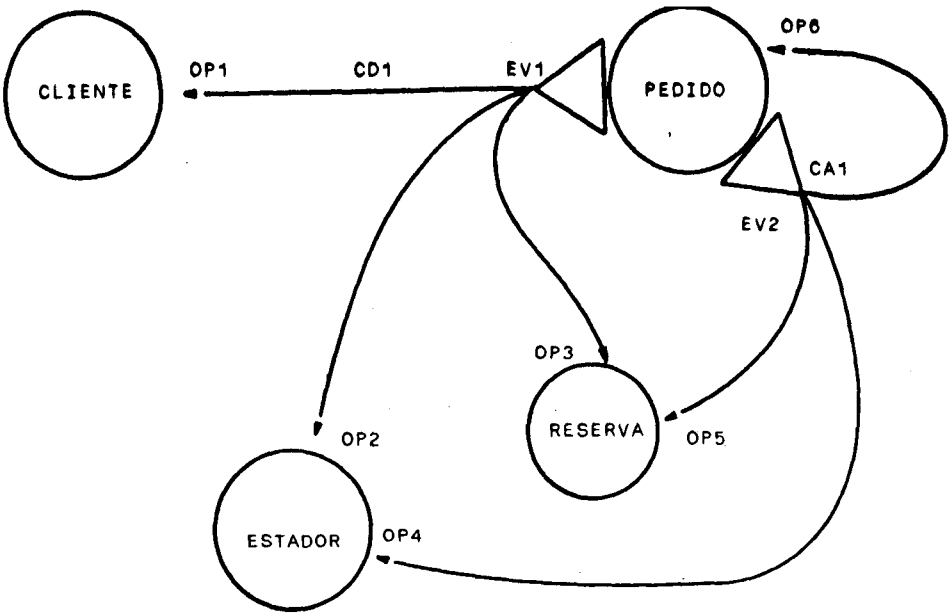
Nocli: numero cliente, entre 80000 e 99999

Noq: numero quarto, entre 0101 e 1010

além dos tipos básicos já conhecidos que são:

Inteiro, Real, Cadeia (String) e Data. Esses tipos servirão na descrição dos atributos.

Assim foi definido o Sub-Esquema Estático.



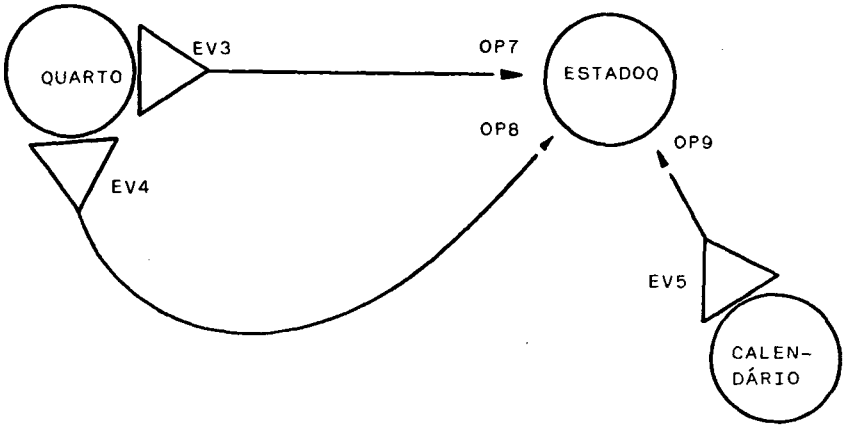




**FIGURA 11:** *Diagrama da Dinâmica manual do exemplo: "caso de reserva em hotel".*

Definição do Sub-Esquema Dinâmico:

Em torno do Sub-Esquema Estático, será construída a parte dinâmica do sistema. Isto é, serão definidos os eventos e as ações. Ver o diagrama da dinâmica na **figura 11.**

Eventos:

EV1: inclusão de um pedido de reserva

EV2: inclusão de um pedido de cancelamento

EV3: criação de um novo quarto

EV4: supressão de um quarto

EV5: mudança de dia do calendário

Ações:

OP1: incluir novo cliente

OP2: registrar a reserva

OP3: registrar o quarto em nome do cliente

OP4: alterar a reserva

OP5: desfazer a reserva do quarto

OP6: excluir o registro do pedido de reserva

OP7: incluir quarto novo

OP8: alterar estado do quarto

OP9: atualizar estado do quarto

Condições de aceitação:

CA1: se o cliente for novo

Condições de disparos:

CD1: se o pedido ja tinha sido feito

observações:

1) A cada operação, evento ou condição é associado um texto expresso numa linguagem formal ( a lingugem MIND-BFB na macrometodologia MIND-BFB, a ISDEL na REMORA). Exemplos destes textos serão mostrados no item 4.5.

2) É lembrado que no estagio atual da macrometodologia MIND-BFB só foi proposta a metodologia de especificação do problema e para essa foi adotada a metodologia REMORA com algumas alterações (expostas no item 3.1.). A modelagem lógica padrão da REMORA se justifica pela grande elementaridade dos c-elementos (3FNP). No contexto da MIND-BFB, a passagem do nível conceitual ao físico dependerá da utilização específica dos usuários (onde serão considerados aspectos como sincronismo, por exemplo) e os parâmetros técnicos. Neste exemplo então, como no resto do trabalho, só há preocupação com o nível conceitual.

## 3.3. CONCLUSÃO:

A macrometodologia MIND-BFB é a integração de três metodologias: Controle de Projeto, Especificação do Problema e Construção da Solução. Para a Especificação do Problema foi adotada a metodologia REMORA com certas alterações. A REMORA é uma metodologia que subdivide a especificação do sistemas em três níveis: Um nível descritivo, que modela as informações coletadas sem preocupação com redundância, incompleteza, etc. Em seguida é feita uma modelagem conceitual baseada nos conceitos da metodologia (normalizações, controles de integridade, etc.) e que vai permitir a obtenção de uma especificação mínima, completa e fiel. A modelagem conceitual não considera aspectos como sincronismo, retrições físicas da instalação e modo de uso do sistema. Esses aspectos passam a ser considerados na modelagem interna que é concluída com a implementação física do sistema. É bom lembrar que a metodologia REMORA dá ênfase maior à estruturação dos dados (Sub-Esquema Estatico) cuja especificação precede a definição dos eventos e das ações.

# 4. PROPOSTA DE UMA FERRAMENTA DE APOIO: MIND-AID

## 4.1. INTRODUÇÃO:

Foi visto que o Esquema Conceitual é uma expressão abstrata do fenômeno real a ser representado no SI. Construir um SI a partir de um EC é só possivel se esse último for correto. As qualidades que o EC deve possuir ajudam no controle dessa descrição conceitual. Elas são fidelidade, consistência e completeza.

Foi proposto aqui uma ferramenta automatizada, a MIND-AID, que possa ajudar um projetista na definição, controle e análise da especificação conceitual.

Essa ferramenta serve para apoiar certos aspectos da metodologia, deixando ao projetista todas as decisões heurísticas exigidas pelo método.

Referindo-se à classificação dos CASEs no capitulo 1, esta ferramenta se enquadra no grupo de CASEs de Apoio a Metodologias que são centrados em torno de metodologias (no caso, a REMORA). Nos itens a seguir, serão definidos os objetivos básicos perseguidos na construção da ferramenta, a sua arquitetura geral, uma descrição detalhada dos diferentes módulos que compõem a ferramenta e, no final, a solução através do CASE do exemplo didático do caso de reserva em hotel que foi tratado manualmente no capitulo anterior. Um conhecimento mais detalhado e o modo de operação da ferramenta poderão ser encontrados no manual [YAY 88], editado com um exemplo e disponível numa outra encadernação.

## 4.2. OBJETIVOS:

A ferramenta proposta assiste o projetista no nível conceitual da especificação da metodologia. O nível conceitual corresponde às fases de Ante-Projeto e Concepção-Funcional da macrometodologia descrita.

As ferramentas são recursos técnicos que dispomos para a modelagem do problema e sua descrição através da linguagem [ROL 83]. A praticidade de uma metodologia depende das ferramentas de apoio disponíveis. Esta ferramenta automatizada deve assegurar a passagem de um Esquema Conceitual a um SI concreto, implementável num dado ambiente técnico. Para que o SI continue fiel à realidade no decurso do tempo, é necessário que a ferramenta gerencie sua evolução paralelamente e conforme a evolução da realidade. Para isso, a ferramenta deve:

- Permitir ao projetista representar os resultados da sua análise com os conceitos do modelo e o formalismo da linguagem.

- Controlar as coerências sintáticas e semânticas da representação.

- Possibilitar e controlar modificações de representação e informar o projetista das consequências.

- Permitir uma utilização diversificada, flexível e simples da representação.

São esses quatro objetivos que orientaram a construção da ferramenta.

## 4.3. ARQUITETURA GERAL:

### 4.3.1. Introdução:

Será apresentado aqui a linguagem utilizada para a implementação da ferramenta, PROLOG [BOR 86], e um conceito de estrutura de dados (tipos abstratos) que influenciou a área de programação e que foi adaptado ao modelo REMORA por [BAR2 87]. Essa apresentação é necessária pois a linguagem e o conceito caracterizam a ferramenta.

. linguagem Prolog:

Prolog é uma linguagem bastante utilizada, com facilidade para

- Produção de protótipos de programas
- Implementação de Bases de Dados Relacionais
- Processamento de linguagens naturais
- Construção de interfaces de linguagens naturais para sistemas já existentes
- Controle de processos
- Etc..

Um programa PROLOG pode ser visto como:
- Uma especificação de fatos
- Uma especificação de regras sobre esses fatos e suas relações
- Consultas sobre os fatos e suas relações

De um lado, PROLOG se revela muito adequado para descrever Bases de Dados. A utilização dessa linguagem lógica acaba com a diferença que se faz entre dados e programas (fatos e regras são misturados). As mesmas estratégias aplicadas na resolução dos programas se aplica também para consultar os dados [BOR 86].

Já que a Base de Dados pode ser vista como um programa PROLOG, a restituição é efetuada através de uma resolução. Por exemplo, seja um predicado, P, de grau n. A restituição do j-ésimo argumento corresponde a valores específicos dos outros argumentos é obtida perguntando-se :

```
?- P(v1,v2,...,vj-1,X,vj+1,....vn).
```

Pode-se ter uma ou várias respostas, dependendo do conteúdo da Base de Dados.

Por outro lado, um programa numa linguagem de terceira geração (FORTRAN, PASCAL, C) é uma tradução de um algoritmo, isto é, um método de resolução de um problema: são linguagens imperativas, procedimentais. O programador especifica **COMO** resolver o problema. Ao contrario, um programa em PROLOG só enuncia fatos e regras relativos ao problema: é uma linguagem declarativa. A estrutura de dados LISTA é bastante utilizada em PROLOG, o que facilita a manipulação das estruturas de dados (tabela, fila, pilha, arvore, etc.). Exemplos de listas são:

[joão, maria, alfredo]
[a|Y] lista que tem como cabeca "a" e o resto é Y
[] lista vazia

Por essas razões, foi decidido usar Prolog como linguagem de implementação de MIND-AID pois precisamos, na nossa metodologia, de um ambiente de desenvolvimento que permita :

- Representar as estruturas relacionais resultantes da nossa análise, formando uma Base de Dados relacional
- Implementar primitivas de manipulação dessas estruturas
- Fazer o "parsing" dos textos das operações e condições de disparos
- Todas as facilidades de controle de tipos e de integridades que se fazem necessárias

Prolog tem uma sintaxe simples ,mínima e é poderosa [RIN 88]. O compilador que será usado aqui é uma implementação da Borlands.

. introdução dos tipos abstratos:

Queremos um SI formalizado de tal maneira que possa ser possível determinar o conjunto de fórmulas verdadeiras, bem formadas na especificação, como é o caso em certas linguagens de programação. Os termos válidos as especificações obtidas baseado no modelo de [THI 85].

Em relação ao nosso modelo, pode-se considerar que o SI é um produto cartesiano de objetos,eventos e acões. As restricões de integridade são expressas através de predicados ao longo da modelagem.

Notando esse fato :

SI = < objetos:c-objeto,eventos:c-evento,acões:c-ação>,

pode-se definir cada um desses três tipos como produtos cartesianos de outros tipos a um nível mais baixo:

c-objeto = <atribobj1:tipob1,...,atribobjn:tipobn>

c-evento = <atribevent1:tipev1,....,atribevn:tipevn>

c-ação = <atriação1:tipac1,.....,atriação:tipevn>

Cada um dos tipos tipobj,tipevj e tipacj é definido através de tipos básicos ( inteiro, texto,etc.... ) e um conjunto de tipos de base estruturados.

Então, o sistema estará obrigando o projetista a dar uma descrição sintaticamente correta das estruturas de dados ( os textos serão referenciados e suas especificões serão verificadas em outro momento).

## 4.3.2. A Base de Dados da ferramenta:

Os diferentes módulos que compõem a ferramenta são integrados ao nível da BDD resultando da especificação progressiva. É uma BDD Prolog (então Relacional) que vai conter os elementos do modelo (tipos, atributos, objetos, eventos, ações, constatações, disparos, modificações e relacionamentos de objetos). Um exemplo de Base de Dados de especificação encontra-se no anexo 4; esse exemplo é relativo ao caso de hospedagem em hotel tratado nos itens 3.2.5. e 4.5. Os esquemas comentados destas estruturas são os seguintes:

tipo(<nome-tipo>,<estrutura>,<valores>,<domínio>,<comentário>)

atributo(<nome-atributo>,<tipo>,<validação>,<comentário>)

c_objeto(<código-objeto>,<nome-objeto>,<lista de atributos>, <comentário>,<data-criação>,<data-supressão>)

Para cada C-OBJETO é definido um código, um nome e uma lista de atributos. Cada atributo é por sua vez especificado por um nome, um tipo, uma regra de validação e um comentário. Os tipos permitem generalizar a definição dos atributos. Essa estrutura lembra linguagens como Pascal, Modula-2, etc...

c_relacionamento(<nome-relacion.>,<lista de atributos>, <nome-regra>,<comentário>,<data-criação>,<data-supressão>)

O c-relacionamento é um objeto particular cuja chave e composta das chaves de pelo menos dois outros objetos.

c_evento(<código-evento>,<nome-evento>,<comentário>, <data-criação>,<data-supressão>)

Estes cinco parâmetros bastam para definir um evento isolado.

c-constata(<código-evento>,<código-objeto>,<condição-aceitação>, <lista de objetos-consultados> <data-criação>,<data-supressão>)

A associação é especificada pelas referências dos dois c-elementos, de um predicado condicionando o evento e de uma lista de objetos a serem consultados durante a avaliação do predicado.

c_ação(<código-ação>,<nome-ação>,<tipo-ação>,<objeto-afetado>,
<lista de objetos-consultados>,<lógica>,<comentário>,
<data-criação>,<data-supressão>)

O Esquema de uma c-ação é composta do código, do nome, do nome do tipo, do objeto afetado pela ação, de uma lista de objetos consultados, da referência do texto correspondente (lógica) e de um comentário reduzido lembrando o seu papel.

obs: qualquer referência a um c-elemento no esquema de um outro é materializado pelo código do primeiro.

c_dispara(<código-evento>,<código-ação>,<código-obj-afetado>,
<condição-execução>,<lista de objetos-consultados>
<fator disparo>,<lista de objetos-consultados>
<data-criação>,<data-supressão>)

Três c-elementos são envolvidos nessa associação: o evento cuja constatação implica o disparo de uma ação e que vai modificar um objeto. Além das três referências, é necessário no esquema de C-DISPARA uma condição de execução ( predicado ), um fator de aplicação (predicado) e as listas de objetos a serem consultados para avaliação dos dois predicados.

c_modifica(<código-ação>,<código-obj-afetado>,
<data-criação>,<data-supressão>)

Essa associação indica qual objeto é alterado por qual ação.

### 4.3.3. Interface com o usuário:

Para facilitar ao máximo o uso da ferramenta e graças às facilidades encontradas no TOOLBOX [BOR 87], foi conseguido um sistema flexível e bastante amigável baseado em menus e janelas. Para se ter uma idéia das facilidades oferecidas ao usuário, são citados a seguir alguns exemplos:

Linha de status: é uma mensagem na última linha da tela que informa dos procedimentos a serem seguidos ou que avisa os erros cometidos.

Pedidos de nomes de arquivos: Será pedido o nome das Bases de Dados da sua especificação ou de um texto de operação ou de restrição, expressa na linguagem MIND-BFB. Haverá um nome e uma extensão "default" numa janela de uma linha. Naturalmente, o usuário pode não aceitar nenhum dos dois e teclar o nome desejado. Nesse caso, pode-se especificar outro diretório. Senão

é só teclar <ENTER>. Outra possibilidade é remover o conteúdo da janela do nome e teclar <ENTER>, aparecerá numa janela volátil a lista dos arquivos com a extensão "default" do diretório em uso.

Especificação dos textos: telas formatadas com controles de consistências permitem entrar os elementos do modelo. Mas, como foi anunciado acima, o modelo é completado por textos (das operações, de integridade, de comentários). São oferecidos ambientes de edição em várias fases da especificação para a digitação destes textos com opção de verificação sintática quando for um texto formal.

Ajuda de contexto: Para cada campo das telas formatadas existe um texto de ajuda lembrando ao usuário os conceitos e as restrições da especificação a esse nível.

Janelas sobre a BDD: Várias teclas foram programadas para que a qualquer momento o usuário possa ver numa janela volátil o que existe especificado na BDD Prolog para um dado esquema (tipo, objeto, evento, etc...).

Opções: A cada vez que o usuário precisar escolher entre várias opções, é apresentado a ele uma janela de múltipla escolhas entre as quais ele navega com as setas do teclado.

### 4.3.4. Processos:

A modelagem conceitual é a descrição semântica do mundo real. O resultado dessa modelagem é uma representação formal do sistema sob os aspectos estáticos e dinâmicos.

Ela é

a) obtida através de um modelo construído usando-se conceitos teóricos e regras metodológicas.

b) expressa através de uma linguagem formal [FOU 82].

Por isso a ferramenta deve assegurar meios de especificar o sistema respeitando os conceitos do modelo e dar suporte para a linguagem formal definida. Então as características do CASE proposto para ajuda à concepção são:

- integração da especificação
- verificação
- documentação

O sistema vai guiar o projetista e fazer certas deducões mas não vai subtitui-lo totalmente : o projetista permanece como tomador de decisões durante a análise. É importante perceber que foi privilegiada a descrição dos objetos (parte estática) em relação aos ciclos(parte dinâmica).

Os passos do uso da ferramenta são:

a) definição do esquema conceitual:

- criação de um objeto e seus constituintes

- criação do Sub-Esquema dinâmico (com base nos objetos definidos)
- supressão de um objeto (em caso de modificação)

- modificação de um ciclo

b) controle do esquema conceitual:

- são efetuados vários controles na definição do esquema conceitual.

- uma serie de programas extras procederão a certas verificacões, com processamento por lotes.

c) documentação:

- edição Sub-Esquema de dados
- edição Sub-Esquema da dinâmica
- edição chaves das relacões
- várias outras edicões através de primitivas previstas para usuário final no sistema ou através das primitivas do próprio prolog, o que oferece bem mais possibilidades.

Os módulos implementados são os seguintes.

MC1 : criação Sub-Esquema estático

(entrada da descrição e normalização dos tipos,c-objetos e c-relacionamentos)

MC2 : documentação Sub-Esquema estático

(relatórios sobre o c-objetos,atributos,tipos,etc)

MC3: criação Sub-Esquema dinâmico

(criação dos c-eventos,c-acões,c-dispara,etc)

MC4: documentação Sub-Esquema conceitual

(relatórios sobre diagrama da dinâmica,ciclos,etc)

MC5: modificacões Sub-Esquema estático

(modificações de c-objetos,atributos,etc)

MC6: modificacões Sub-Esquema dinâmico

(modificacões de ações,ciclos)

MC7: controles de integridade do esquema conceitual

(programa em lotes para submeter a descrição conceitual ás regras de integridade)

MIND_BFB: compilação textos

("parser" ou "analisador" para verificação da especificação dos textos das acões e predicados)

SORT: ordenação alfabética das ocorrências dos esquemas das relações da BDD Prolog da especificação

TROBJ: conversão das esquemas das relações para estruturas Dbase III plus

TRCON: transferência da BDD Prolog da especificação conceitual para Dbase III plus

## 4.4. DESCRIÇÃO DETALHADA DOS MÓDULOS:

### MC1 = CRIAÇÃO Sub-ESQUEMA ESTÁTICO:

Uma vez chamado este módulo, o sistema pede ao projetista para:

- definir os tipos de dados que vão ser utilizados na descrição dos c-objetos. Relacões prolog são criadas com o predicado tipo para representar esses tipos na METABASE.

Depois, o sistema pede a relação Ro (relação original) que ele deseja representar ( correspondente a um objeto do modelo descritivo). Isto é, o conjunto dos atributos do objeto descrito.

O sistema usa o algoritmo de normalização, apresentado acima, de normalização de relacões na Terceira Forma Normal (3FN) e que, dada uma representação

$$So = ( Ro (A), D, Fo )$$

leva a uma final

$$Sf = \cup Si = (Ri (Ai), Di, Fi ), i=1,n$$

onde cada relação Ri está na 3FNP.

A : conjunto de atributos da relação
D : domínios dos atributos
F : dependências funcionais entre atributos

É bom lembrar que a versão MIND-BFB da REMORA adota a 3FN. Para isso:

- o sistema cria relações prolog de dependências funcionais entre atributos. O módulo analisa essas dependências e cria as relações, Ri, derivadas.

- são criadas esquemas de relações na BASE DE DADOS prolog para representar os atributos encontrados nas relações.

- percebendo o fim da criação dos objetos, o sistema sugere ao projetista a criação dos C-relacionamentos entre objetos. Na verdade os C-relacionamentos já existem entre os objetos deduzidos da normalização. Mas uma função vai varrer a BDD na busca de objetos cujas chaves são compostas da chave de pelo menos dois outros objetos. Nesse caso, são candidatos a C-relacionamentos. E tarefa do usuário confirmar ou não essa renomeação daqueles objetos.

A listagem comentada deste módulo pode ser encontrada no anexo 3.

MC2 - DOCUMENTAÇÃO Sub-ESQUEMA ESTÁTICO :

Como foi dito acima, foi privilegiado o aspecto estático do modelo. O que explica a necessidade de se ter um módulo só para se obter uma documentação do Sub-Esquema estático. A importância dessa opção será percebida na apresentação do método.

Através deste módulo são propostas quatro funções de documentação sobre o Sub-Esquema estático.

Essas funções parecem essenciais para a construção do Sub-Esquema dinâmico. Também a documentação assim gerada servirá muito bem para outras necessidades (usuários, programadores, elaboração esquema lógico, etc...).

O usuário dispõe de cinco tipos de documentação sobre a BDD a esse nível que são:

. a relação dos tipos
. a relação dos atributos
. a relação dos objetos deduzidos da normalização
. a relação dos relacionamentos deduzidos da busca de relacionamentos
. a relação das chaves deduzidas da normalização

O usuário decidirá da unidade de saída (tela ou impressora).

MC3 - CRIAÇÃO Sub-ESQUEMA DINÂMICO :

Este módulo permite criar o Sub-Esquema dinâmico (eventos, ações e seus relacionamentos) acima do Sub-Esquema estático definido no MC1. O módulo monta uma lista de todos os objetos existentes na BDD. Para cada objeto, ele exibe uma tela formatada de definição de evento constatado a partir deste objeto. Enquanto houver eventos a serem definidos acima daquele objeto, ele aparecerá numa nova tela formatada, senão, o sistema repetirá o mesmo processo para o objeto seguinte da lista. Uma vez definidos os eventos, a segunda opção é usada para definir as ações do mesmo jeito: uma lista de eventos existentes na BDD é montada e para cada evento, o módulo pedirá para definir todas as ações que ele dispara. Quando um evento é definido, o módulo gera automaticamente o c-constata e quando uma ação é definida, são geradas o c-dispara e o c-modifica associadas.

MC4 - DOCUMENTAÇÃO ESQUEMA CONCEITUAL :

Uma vez construído o Sub-Esquema dinâmico, dispomos de toda a parte estrutural do esquema conceitual. Só faltam os textos das ações e dos predicados. Mas foi considerado o tratamento dos textos em separado. São propostas aqui funções que permitem entender e obter visões particulares do esquema conceitual:

. relação dos eventos
. relação das constatações
. relação das ações
. relação dos disparos
. relação das modificações
. diagrama da dinâmica: esboço linear do diagrama da dinâmica, a partir do qual pode-se construir um diagrama mais compacto a mão.
. correspondências entre nomes descritivos e nomes de acesso (códigos). Isso é necessário, já que a menos das suas estruturas específicas, cada objeto, evento ou ação é referênciado pelo seu código no resto da especificação.

MC5 - MODIFICAÇÃO Sub-ESQUEMA ESTÁTICO :

Tanto a metodologia REMORA como a MIND-BFB dão ênfase ao Sub-Esquema estático (que representa a estrutura de dados através dos tipos, atributos e objetos). É considerando essa opção que foi proposto este módulo. É assumido que qualquer alteração no Sub-Esquema estático implica a redefinição da parte dinâmica. Entretanto, a tão desejável característica de "traceability" existe a esse nível. Isto é, a modificação de uma característica de um tipo implica a mesma nos atributos e objetos dependentes e a modificação de um atributo é igualmente refletida nos objetos dependentes. Por isso, são oferecidas aqui três primitivas de modificação, que são:

. modificação de um tipo
. modificação de um atributo
. supressão de um objeto

MC6 - MODIFICAÇÃO Sub-ESQUEMA DINÂMICO:

O SI é uma abstração da organização. Mas a organização evolue na sua estrutura e no seu funcionamento. O SI tem que acompanhar essa evolução. Uma regra de gestão pode ser modificada sem que seja necessário modificar a estrutura de dados.

Oferecemos neste módulo quatro funções de modificação dessas regras. Essas modificações se baseiam no ciclo, pois uma regra é definida por um evento associado a um objeto e um conjunto de ações.

As cinco funções são:

- supressão de um evento
- criação de um evento
- criação de uma ação
- modificação de um evento
- modificação de uma ação

Quando o usuário aciona a função de supressão de um evento , ele recebe uma tela formatada vazia e ao digitar o código do evento seguido da tecla <F10>, ele recebe a tela preenchida com todas as características do evento, daí ele pode confirmar ou desistir da supressão do evento. A supressão de um evento implica na supressão da relação constata associada. A criação de um elemento é aceita pelo sistema depois de verificar que ele já não existe e quando o usuário quer modificar um elemento que não existe ele recebe uma mensagem de erro.

MC7 - INTEGRIDADE DA DESCRIÇÃO :

Um esquema conceitual válido é um esquema conceitual cuja expressão respeita as propriedades dos conceitos do modelo e que possui as qualidades de um esquema conceitual canônico (fiel, coerente, não redundante). Existem duas classes de regras de validade :

- sintaticas: controle do uso correto do modelo
- semânticas: verificar as qualidades da descrição

As regras de validade sobre a coleção de relações que se tem serão expressas através de operadores de projeção,produto cartesiano e seleção do cálculo relacional e que existem na sintaxe de prolog.

- regra-1: completeza da estrutura:

  - qualquer objeto modificado por uma ação ou constatado por um evento é descrito.

    evento[obj] é subconjunto de objeto[obj]

ação[obj] é subconjunto de objeto[obj]

- toda ação ou evento que aparece numa associação "dispara" é descrita e vice-versa,

- todo evento dispara pelo menos uma ação e toda operação é disparada por pelo menos um ação

disparo[op] = ação[op] e dispara[ev] = evento[ev]

. regra-2: ciclos idênticos:

Não existem duas descrições de uma mesma regra de gestão: ( não presença de 2 ciclos idênticos)

não existe (ev1,ev2) tq:

evento[ev1,obj] = evento[ev2,obj]
e,
dispara[ev1,op1,cond1] = dispara[ev2,op1,cond1]

. fidelidade dos nós terminais :

Um nó terminal do diagrama da dinâmica não tem nenhum evento associado. O que significa que as mudanças de estado dele não tem nenhuma consequência:

Um nó, o1, é terminal se :

o1 pertence a objeto[código-objeto],
o1 pertence a ação[obj] e,
o1 não pertence a evento[obj].

O projetista é informado só através do relatório, é tarefa dele conferir se corresponde ou não à realidade descrita.

**MIND-BFB:**

"MIND_BFB" é o nome da metodologia, mas também da linguagem formal de especificação que foi adotada para os textos das c-operações, das condições de aceitação dos c-eventos, das condições de disparo das c-operações e dos textos de fator de disparo. Este módulo tem por objetivo oferecer um ambiente de edição e de verificação da especificação dos textos citados.

O MIND-BFB pode ser usado também para editar textos de comentários em sintaxe livre, já que ele oferece um ambiente de edição próprio. Ele oferece um ambiente de manipulação dos textos. Pode ser usado então, para criar o arquivo-texto,

carregar, salvar ou compilar os textos. A gramática da linguagem encontra-se em anexo. Entretanto, porque uma linguagem formal de especificação apesar das estruturas relacionais ?

Para obter uma representação do SI suficientemente formalizada, é necessário completar o Esquema Conceitual por um sistema de integridade formalmente expresso:

especificação conceitual = EC + sistema de integridade

Qualquer modelagem possui simplificacões e limites na representação da realidade. Subsistem aspectos da realidade que o modelo não pode representar integralmente [FOU 82].

Daí, Sistema de Integridade = complemento de ordem semântica ao EC escrito sob forma de restrições de integridade. Como é o caso das acões, os predicados representando as restrições de integridade serão expressos através de textos. Por isso foi procurado implementar uma linguagem formal de especificação.

A linguagem de especificação deve ser baseada no modelo escolhido e a lógica pode ser considerada como uma extensão natural do modelo relacional (usado na REMORA), pois cada relação pode ser expressa como um predicado da forma P(t1, ..., tm). A linguagem deve, então, permitir uma especificação abstrata, estruturada e formal da realidade. Ela deve ser completa e de alto nível. Ela deve ser um bom meio de comunicação (entre usuários, projetista e programadores) e para isso, deve ser declarativa e não procedural. Ela deve dispôr de uma sintaxe quase livre e com facilidades de um ambiente de programação avançado. Ela deve ser facilmente convertível para outra linguagem sem deixar de ser de alto nível.

A linguagem deve permitir díalogo fácil com o usuário durante a fase de análise e ficar o mais independente possível de restrições de implementação.

Uma linguagem de especificação (que não vai produzir código diretamente executado pela máquina) deve ser a mais independente possível de critérios de implementação ou otimização, o importante sendo a comunicação com o usuário final e o grupo de implementação. Essas considerações guiaram na escolha da síntaxe da linguagem e a lógica é aquela que se aproxima mais destes conceitos.

Um programa lógico é um conjunto de clausulas da forma :
P0 <--- P1, P2, ... , Pn.
Cada Pi é um literal e possui a forma P(t1,t2,...,tm), onde P é o predicado e os ti são os termos. P0 é a cabeça ou a conclusão da clausula e P1 a Pn formam o corpo e são as condições da clausula [COH 88] e [KOW 79]. A interpretação semântica sería: P0 é verdadeiro se P1 e P2 e ... e Pn são verdadeiros. A primeira atração da lógica é o elegante formalismo usado para representar

fatos (dados), consultas e regras (informações dedutíveis) numa forma única. Por exemplo, " João é o pai de Maria" é representado:

pai(joao, maria) <---.

Para representar a consulta: "Quém é o pai de Maria?", usa-se:

<--- pai(X, maria), onde X é uma variável que vai ser subtituída por joão (esse processo é chamado de casamento). A mesma forma, simples e natural, é assim usada para representar tanto os dados como as consultas. A maior contribuição da lógica para a área das Bases de Dados é a introdução de informações dedutívas através de regras. Em várias situações, é vantajoso representar certos fatos através de implicações lógicas em vez de uma lista exaustiva de fatos. Por exemplo, para representar a relação "avô", é possível só definir uma regra acima das relações "pai" da seguinte maneira:

avô (X,Y) <--- pai(X,Z), pai(Z,Y).

Com construções desse tipo, deixa de ser necessário armazenar individualmente todas as relações "avô", a máquina de inferência se encarrega de proceder aos casamentos devidos varrendo a Base de Dados [RIN 88].

As vantagens são que detalhes de implementação são transparentes ao usuário, permitindo seu acordo sobre algo que entendeu; menor código mas maior poder de expressão são conseguidos e ficam minimizados os problemas de comunicação analistas-programadores.

Foi então construído um "analisador" sintático ou ("parser") da linguagem lógica proposta. Um "analisador" é um programa que é capaz de reconhecer a estrutura de um texto. Por exemplo, o compilador Pascal contém um "analisador" que reconhece as instruções IF, WHILE, REPEAT, CASE, etc. Outra função de um parser é de transformar o texto fonte numa forma adequada para as fases seguintes de uma compilação ou de uma interpretação. As estruturas a serem reconhecidas por um parser são normalmente descritas por uma gramática. Uma gramática pode ser descrita de várias maneiras. O TOOLBOX usa o Backus Naur Form (BNF). Depois de determinar que uma sentença está de acordo com as regras da gramática original, uma árvore é construída para mostrar a estrutura do texto. A descrição da gramática não é feita com a notação BNF padrão. Existem três tipos de produções que definem como construir as estruturas gramaticais umas a partir das outras:

- Lista_de__produção ("production list") é um nome de produção seguido de + (uma ou várias ocorrências) ou * (zero ou várias ocorrências). Por exemplo, LISTPARAM = PARAM*.

- Produção geral ("general production") é composto de um ou vários grupos de produções individuais, cada um tendo a mesma prioridade no mesmo grupo. Os grupos de prioridade são separados por "--" enquanto que as produções individuais o são por "," em cada grupo.

- Produção individual ("individual production") é um nome de produção seguido de uma lista de zero ou vários parâmetros.

Cada produção é seguida da especificação do termo prolog correspondente.

Outras diferenças superficiais são a eliminação de "<" , ">" e "::=" subtituído por "=".

O gerador de "analisador" resolve muitos problemas semânticos através a sintaxe da sua gramática que o BNF padrão não permite:

Precedência: Os signos "--" separam os grupos de prioridade. Assim, numa produção especificando, por exemplo, as expressões de cálculo, tem-se a adição e a subtração no mesmo grupo enquanto que a multiplicação e a divisão ficarão num mesmo grupo de maior prioridade.

Associatividade: Por "defaut", tem-se a associatividade pela esquerda. Poe exemplo, em 2+3+4, será feito 2+3 e depois 5+4. Mas se for necessário implementar uma operação associativa pela direita, tem que especifica-lo pela palavra-chave "rightassoc". O que é necessário para, por exemplo, uma operação como 2^3^4.

Ambiguidade: Às vezes, existem mais de uma interpretação possível de uma expressão para uma gramática dada. É difícil, por exemplo, ter-se uma interpretação única da seguinte instrução:

```
se c1 então
 se c2 então op1
  senão op2
```

O parser não vai saber, a priori, se é

se c1 então ( se c2 então op1 senão op2) ou

se c1 então ( se c2 então op1) senão op2.

Esse problema é implicitamene resolvido, nessa gramática, pela ordem das produções correspondentes.

**SORT:** No caso de se desejar, por razões de praticidade, obter relatóros da especificação com os elementos do modelo (objetos, ações, eventos) na ordem alfabética, deve ser rodado o programa SORT com a Base de Dados.

**IROBJ:** A MIND-AID é destinada a assistir durante a especificação conceitual, isto é, o resultado obtido poderá sofrer modificações nos níveis lógico e físico que deverão seguir normalmente.

Mas pode-se desejar implementar fisicamente esse primeiro resultado, pelo menos a estrutura de dados. Essa escolha é bastante razoável, já que os objetos estão normalizados e só até a Terceira Forma Normal (3FN). Nesse caso, a chamada desse módulo com o resultado da especificação do Sub-Esquema Estático (extensão "est") permitirá gerar relatórios das estruturas de cada objeto indicando:

- número de campo (Field)

- nome do campo (FieldName)

- tipo (Type)

- tamanho (Width)

- Restrições

Esse relatório permite praticamente realizar a implementação física das estruturas dos dados diretamente num ambiente relacional. A semelhança com dBASE é nítida até nos nomes dos campos como pode ser visto acima.

Na verdade, é oferecido uma descrição mais rica do que é necessário para o dBASE e outros gerenciadores relacionais: as restrições. Elas poderão ser procedimentos externos na linguagem de manipulação do gerenciador empregado e serem referenciadas dentro dos programas que usam os objetos.

**IRCON:** A linguagem dBASE conheceu uma grande popularidade e, no BFB, qualquer analista ou programador sabe manipula-la. Foi então oferecido o módulo TRCON para converter as Bases de Dados Prolog em Bases de Dados dBASE, possibilitando assim maior facilidade de uso para a maioria dos usuários potenciais.

Evidentemente, não é previsto que os usuários modifiquem estes arquivos, a transferência tem como objetivo facilitar as diferentes formas de consultas para quem utilisa dBASE.

## 4.5. Exemplo: caso de hospedagem em hotel:

A seguir estão listados alguns relatórios gerados através do uso da ferramenta MIND-AID, no caso do exemplo visto em 3.2.5.

Tem-se primeiro as saídas de MC2 que são as listas dos tipos, dos atributos, dos objetos, dos relacionamentos e das chaves. Essa saída do MC2 representa o resultado da especificação do Sub-Esquema Estático. Os objetos foram derivados da normalização efetuada pelo módulo MC1, a partir da especificação das Dependências Funcionais, feita pelo usuário **(saídas 1 a 5).**

Em seguida, tem-se as saídas de MC4 que são as listas dos eventos, das ações, das constatações, dos disparos, das modificações e do diagrama da dinâmica **(saídas 6 a 11)**. Esse conjunto de relatório dá uma visão completa e integrada dos elementos da modelagem conceitual. Também desse módulo, saem três relatórios que lembram as correspondências existentes entre os nomes descritivos (nomes) e os nomes de acessos (codigos) para os três elementos básicos do modelo: objeto, evento e açao **(saída 12)**.

O terceiro tipo de saída é um exemplo de relatório gerado pelo módulo MC7 que informa de eventuais incoerências **(saída 13)**. Seguem a saída do módulo TROBJ **(saída 14)** que, a partir da análise da especificação do Sub-Esquema Conceitual contida na BDD-Prolog, gera a estrutura física de cada objeto. Os esquemas dos objetos assim gerados são diretamente implementáveis em ambientes relacionais com, a mais, indicações sobre restrições de integridade relativas aos atributos. Por fim, tem-se as saídas do TRCON **(saída 15)**. Esse módulo transforma a BDD prolog da especificação conceitual em várias BDD Dbase III plus, uma para cada elemento. As listagens representam o conteúdo dessas BDD Dbase.

Seguem alguns textos, expressos na linguagem formal MIND-BFB, para eventos, ações e condições **(saída 16)**.

T I P O S

| nome | estrutura | coment | validacao | valores(em baixo) | |
|---|---|---|---|---|---|
| DATA | simples | nil | regra(R-DATA) | | nil |
| INTEIRO | simples | nil | nil | | nil |
| CADEIA | simples | nil | nil | | nil |
| NOQ | simples | COM-NOQ | intervalo(0101..1010) | | N10 |
| NOCLI | simples | COM-NOCLI | intervalo(80000..99999) | | N5 |
| TIPOQ | simples | nil | valores("SIMPLES","MEDIO","LUXO") | | A10 |
| TIPOR A10 | simples | nil | valores('SIMPLES','MEDIO','LUXO','CANCEL") | | |
| NOME | simples | nil | nil | | A30 |

SAIDA 1

A T R I B U T O S :

SAIDA 2

| nome | tipo | validacao | coment |
|---|---|---|---|
| ESTADO | CADEIA | valores('Livre','Ocupado') | nil |
| DATAHOQ | DATA | nil | nil |
| TIPOQ | TIPOQ | nil | nil |
| PREGOQ | INTEIRO | nil | nil |
| DURACAO | INTEIRO | nil | nil |
| CUSTO | INTEIRO | regra(CUSTO) | nil |
| NOQ | NOQ | tabela(QuartDisp) | nil |
| ESTADOP | CADEIA | valores("FEITA","CANCELADA") | nil |
| DATAMOR | DATA | nil | nil |
| INICIOP | DATA | nil | nil |
| DATAFIM | DATA | nil | nil |
| DATAINICIO | DATA | nil | nil |
| NOCLI | NOCLI | nil | nil |
| NOMECLI | NOME | nil | nil |
| ENDERCLI | NOME | nil | nil |
| TIPOR | TIPOR | nil | nil |
| DIA | DATA | nil | nil |

C_O B J E T O S

SAÍDA 3

| codiso | nome | criacao | supressao | atributos |
|---|---|---|---|---|
| 081 | QUARTO | 881117 | 0 | NOQ |
| | | | | TIPOQ |
| | | | | PRECOQ |
| 082 | ESTADOQ | 881117 | 0 | NOQ,ESTADOQ |
| | | | | DATAMUDQ |
| 083 | RESERVA | 881117 | 0 | INICIOR, NOCLI, NOQ |
| | | | | DURACAO |
| | | | | CUSTO |
| 084 | ESTADOR | 881117 | 0 | NOCLI,INICIOR |
| | | | | NOQ |
| | | | | ESTADOR DATAMUDR |
| 085 | CLIENTE | 881117 | 0 | NOCLI |
| | | | | NOMECLI |
| | | | | ENDERCLI |
| 086 | PEDIDO | 881117 | 0 | DATAINICIO,NOCLI |
| | | | | DATAFIM |
| | | | | NOMECLI ENDERCLI |
| | | | | TIPOR |
| 087 | CALENDARIO | 881117 | 0 | DIA |

C_RELACIONAMENTOS : **SAÍDA 4**

| nome | codigo | regra | criacao | supressao | atributos |
|---|---|---|---|---|---|
| I33 | RESERVA | R1 | 881117 | 0 | |
| INICIOR, NOCLI, NOG | | | | | |

DURACAO

CUSTO

SAIDA 5

C H A V E S :

| nome-objeto : | atributos-chaves : |
|---|---|
| ESTADOQ | NOQ,ESTADOQ |
| QUARTO | NOQ |
| RESERVA | INICIOR,NOCLI,NOQ |
| ESTADOR | INICIOR,NOCLI |
| CLIENTE | NOCLI |
| PEDIDO | DATAINICIO,NOCLI |
| CALENDARIO | DIA |

C-EVENTOS

SAÍDA 6

| codigo : | nome : | coment : | criacao | supressao |
|---|---|---|---|---|
| EV3 | CriacaoNovoQuarto | nil | 881117 | 0 |
| EV4 | SupressaoQuarto | nil | 881117 | 0 |
| EV1 | InclusaoPedido | nil | 881117 | 0 |
| EV2 | CancelPedido | nil | 881117 | 0 |
| EV5 | MudancaDia | nil | 881117 | 0 |

C-ACOES

| nome | tipo | alvo | consultados | texto | coment | criacao | supressao |
|---|---|---|---|---|---|---|---|
| IncluirQuartoNovo | inclusao | OB2 | OB1 | | LOP7 | nil | 881117 0 |
| AlterarEstadoQuarto | supressao | OB2 | OB1 | | LOP8 | nil | 881117 0 |
| AlterarEstadoQuarto | alteracao | OB2 | OB2 OB3 OB4 | | LOP9 | nil | 881117 0 |
| IncluirClienteNovo | inclusao | OB5 | OB5 OB6 | | LOP1 | nil | 881117 0 |
| RegistrarReserva | inclusao | OB1 | OB6 | | LPO2 | nil | 881117 0 |
| RegistrarCliente | inclusao | OB3 | OB6 | | LOP3 | nil | 881117 0 |
| AlterarReserva | alteracao | OB4 | OB4 OB6 | | LOP4 | nil | 881117 0 |
| DesfazerReservQuarto | supressao | OB3 | OB3 OB6 | | LOP5 | nil | 881117 0 |
| ExcluirPedido | supressao | OB6 | OB6 | | LOP6 | nil | 881117 0 |

SAIDA

C-CONSTATAS

| evento | origem | aceit. | consultados | criacao | supressao | SAIDA |
|---|---|---|---|---|---|---|
| EV3 | OB1 | nil | OB1 | | | |
| EV4 | OB1 | nil | OB1 OB2 | 881117 | 0 | |
| EV1 | OB6 | nil | OB6 | 881117 | 0 | |
| EV2 | OB6 | CA1 | OB6 | 881117 | 0 | |
| EV5 | OB7 | nil | OB7 | 881117 | 0 | |
| | | | | 881117 | 0 | |

C-DISPARAS

SAIDA

| evento | acao | alvo | aceit. | consultados | abrang | consultados | criacao | supressao |
|---|---|---|---|---|---|---|---|---|
| EV3 | OP7 | OB2 | nil | nil | nil | nil | 881117 | 0 |
| EV4 | OP8 | OB2 | nil | nil | nil | nil | 881117 | 0 |
| EV5 | OP9 | OB2 | nil | nil | nil | nil | 881117 | 0 |
| EV1 | OP1 | OB5 | CD1 | OB5 OB6 | nil | OB7 | 881117 | 0 |
| EV1 | OP2 | OB4 | nil | nil | nil | nil | 881117 | 0 |
| EV1 | OP3 | OB3 | nil | nil | nil | nil | 881117 | 0 |
| EV2 | OP4 | OB4 | nil | nil | nil | nil | 881117 | 0 |
| EV2 | OP5 | OB3 | nil | nil | nil | nil | 881117 | 0 |
| EV2 | OP6 | OB6 | nil | nil | nil | nil | 881117 | 0 |

C MODIFICAS

SAIDA 10

| acao | alvo | criacao | supressao |
|---|---|---|---|
| OP7 | OB2 | 881117 | 0 |
| OP8 | OB2 | 881117 | 0 |
| OP9 | OB2 | 881117 | 0 |
| OP1 | OB5 | 881117 | 0 |
| OP2 | OB4 | 881117 | 0 |
| OP3 | OB3 | 881117 | 0 |
| OP4 | OB4 | 881117 | 0 |
| OP5 | OB3 | 881117 | 0 |
| OP6 | OB6 | 881117 | 0 |

SAÍDA 11

```
OB1 < - - - EV3 - - - > OP7
                        |
                        |
                        V

                        OB2

OB1 < - - - EV1 - -   > OP8
                        |
                        |
                        V

                        OB2

OB2   ** no terminal **


OB3   ** no terminal **


OB4   ** no terminal **


OB5   ** no terminal **


OB6 < - - - EV1 - - - > OP1 OP2 OP3
                        |
                        |
                        V

                        OB5 OB4 OB3

OB6 < - - - EV2 - - - > OP4 OP5 OP6
                        |
                        |
                        V

                        OB4 OB3 OB6

OB7 < - - - EV5 - - - > OP9
                        |
                        |
```

| codigo-objeto : | nome-objeto : |
|---|---|
| OB1 | QUARTO |
| OB2 | ESTADOQ |
| OB3 | RESERVA |
| OB4 | ESTADOR |
| OB5 | CLIENTE |
| OB6 | PEDIDO |
| OB7 | CALENDARIO |

| codigo-evento | nome-evento : |
|---|---|
| EV3 | CriacaoNovoQuarto |
| EV4 | SupressaoQuarto |
| EV1 | InclusaoPedido |
| EV2 | CancelPedido |
| EV5 | MudarC[illegible] |

| codigo-acao : | nome-acao : |
|---|---|
| OP7 | IncluirQuartoNovo |
| OP8 | AlterarEstadoQuarto |
| OP9 | AlterarEstadoQuarto |
| OP1 | IncluirClienteNovo |
| OP2 | RegistrarReserva |
| OP3 | RegistrarCliente |
| OP4 | AlterarReserva |
| OP5 | DesfazerReservQuarto |
| OP6 | ExcluirPedido |

objetos de associacao c_constata que nao existem :

SAIDA 13

objetos de associacao c_dispara que nao existem :

acoes de associacao c_dispara que nao existem :
os seguintes eventos disparam op3 que nao existe :
ev3
os seguintes eventos disparam op4 que nao existe :
ev3

os eventos de associacao c_dispara que nao existem :
as seguintes acoes sao disparadas por ev3 que nao existe :
op3 op4

os ciclos identicos sao :

os casos de impasse sao :

os ciclos sem nenhum pai sao :

os eventos declarados mas sem cabeiras sao :

as acoes declaradas mas nao disparadas sao :

ESTRUTURA DE QUARTO

| Field | FieldName | Type | Width | Dec | Restrições |
|---|---|---|---|---|---|
| 1 | NQQ | Numeric | 1 | 0 | tabela(QuartDisp)intervalo(0101..1010) |
| 2 | TIPOQ | Character | 10 | | nil valores(\SIMPLES\ ,\MEDIO\ ,\LUXO\) |
| 3 | PRECOQ | Numeric | 2 | 0 | nil nil |

ESTRUTURA DE ESTADOQ

| Field | FieldName | Type | Width | Dec | Restrições |
|---|---|---|---|---|---|
| 1 | NQQ | Numeric | 1 | 0 | tabela(QuartDisp)intervalo(0101..1010) |
| 2 | ESTADOQ | Character | 30 | | valores('Livre','Ocupado')nil |
| 3 | DATAMUDQ | Numeric | | 6 | nil regra(R-DATA) |

ESTRUTURA DE RESERVA

| Field | FieldName | Type | Width | Dec | Restrições |
|---|---|---|---|---|---|
| 1 | INICIOR | Numeric | | 6 | nil regra(R-DATA) |
| 2 | NOCLI | Numeric | | 5 | nil intervalo(80000..99999) |
| 3 | NQQ | Numeric | 1 | 0 | tabela(QuartDisp)intervalo(0101..1010) |
| 4 | DURACAO | Numeric | 2 | 0 | nil nil |
| 5 | CUSTO | Numeric | 2 | 0 | regra(CUSTO)nil |

ESTRUTURA DE ESTADOR

| Field | FieldName | Type | Width | Dec | Restrições |
|---|---|---|---|---|---|
| 1 | NOCLI | Numeric | | 5 | nil intervalo(80000..99999) |
| 2 | INICIOR | Numeric | | 6 | nil regra(R-DATA) |
| 3 | NQQ | Numeric | 1 | 0 | tabela(QuartDisp)intervalo(0101..1010) |
| 4 | ESTADOR | Character | 30 | | valores('FEITA','CANCELADA')nil |
| 5 | DATAMUDR | Numeric | | 6 | nil regra(R-DATA) |

ESTRUTURA DE CLIENTE

ATRIBUTO.DDF

| NR | NOME | TIPO | VALIDACAO | COMENTARIO |
|---|---|---|---|---|
| 1 | ESTADOQ | CADEIA | valores('Livre' | nil |
| 2 | DATAMUDQ | DATA | nil | nil |
| 3 | TIPOQ | TIPOQ | nil | nil |
| 4 | PRECOQ | INTEIRO | nil | nil |
| | DURACAO | INTEIRO | nil | nil |
| 6 | CUSTO | INTEIRO | nil | nil |
| 7 | NOQ | NOQ | range(0,CUSTO) | nil |
| 8 | ESTADOR | CADEIA | tabela(QuartDi, | nil |
| 9 | DATAMUDR | DATA | valores('FEITA' | nil |
| 10 | INICIOR | DATA | nil | nil |
| 11 | DATAFIM | DATA | nil | nil |
| 12 | DATAINICIO | DATA | nil | nil |
| 13 | NOCLI | NOCLI | nil | nil |
| 14 | NOMECLI | NOME | nil | nil |
| 15 | ENDERCLI | NOME | nil | nil |
| 16 | TIPOR | TIPOR | nil | nil |
| 17 | DIA | DATA | nil | nil |

SAIDA 15

SAIDA 16

```
CA1(DataInicio, Nocli) se

pedido(DataInicio, Nocli,_,_,_,T) e

nao (T="CANCEL") e

pedido(DataInicio, Nocli,_,_,_,"CANCEL").
```

/* para que a condicao de aceitacao (CA1) seja satisfeita, precisa verificar que existia um pedido de reserva normal e que o segundo pedido veio para cancelar o primeiro. */

## 4.6. CONCLUSAO:

A MIND-AID, dentro do contexto da MIND-BFB, cobre apenas o nível conceitual da fase de especificação do problema. Referindo-se à classificação dos cases vistos acima, o MIND-AID é um CASE de Apoio a metodologia. Ele permite uma especificação progressiva e é subdividido em módulos, cada um lidando com um aspecto específico da especificação.

Tem-se três módulos que cuidam só da especificação do Sub-Esquema estático: isto está de acordo com o espirito da REMORA, que considera que a descrição do aspecto estático predomina sobre o dinâmico. O primeiro (MC1) permite, sob controles de consistências, fazer a especificação dos tipos, atributos e objetos; o segundo (MC2) gera relatórios a partir do resultado de MC1; o terceiro (MC5) facilita midificações dos elementos, apesar de ser possivel fazer essas modificações diretamente na BDD através, por exemplo, do módulo MIND_BFB.

Três outros módulos tratam do aspecto dinâmico, cuja especificação basela-se nos elementos do modelo estático. Assim, tem-se um primeiro módulo (MC3) através do qual se definem os eventos e as ações; um segundo módulo (MC4) que gera os relatórios integrados dando uma visão completa da especificação conceitual; finalmente, um terceiro módulo (MC6) facilita modificações na BDD (mas essas modificações poderiam também ser feitas através do módulo MIND_BFB).

Dois módulos de uso particular permitem ordenar os elementos básicos do modelo (SORT) e controlar alguns aspectos da especificação quanto à coerência do conjunto (MC7). O ultimo módulo lidando com o aspecto dinâmico (MIND_BFB), além de ser um "analisador", oferece todo um ambiente de edição, que serve tanto para os textos expressos na linguagem formal como para modificações na BDD Prolog e edição de textos livres.

É bom lembrar que o EC assim obtido pode ser transformado num Esquema Lógico Padrão (ver item 3.2.4.) e que a MIND-AID não oferece nenhum apoio a esse nível. Entretanto, existe um módulo (TROBJ) que ja faz uma transformação direta do Sub-Esquema Estático para o nível físico, se for desejado. Essa transformação é aceitável, já que os objetos estão normalizados na 3FN.

# 5. CONCLUSÃO GERAL:

## 5.1. Introdução:

Não se pode adotar uma metodologia qualquer para o desenvolvimento de SI num dado Centro de Processamento de Dados (CPD). Entre outros critérios que devem guiar na escolha podem ser citados o problema da adequação do modelo com o ambiente técnico do CPD (gerenciadores de arquivos, sistemas operacionais, etc.), a natureza das aplicações desenvolvidas (pequenas ou grandes, duração de uso longa ou curta, etc.) e, finalmente, as facilidades e flexibilidades que a própria metodologia oferece para o projetista.

O BFB adotou REMORA como componente de especificação de solução da sua macrometodologia, pois ela leva o projetista a perceber, selecionar e classificar os fenômenos individuais. Ela permite definir regras de comportamento e restrições às quais esses fenômenos são submetidos. O SI resultante do uso de REMORA fornece dois tipos de conhecimento:

- O conhecimento das regras de estrutura e de comportamento do sistema (resultado da modelagem).

- O conhecimento dos fatos reais da organização (dados, programas e comandos).

O SI resultante dessa modelagem tem uma semelhança com os Sistemas Especialistas já que o primeiro tipo de conhecimento pode ser visto como a Base de Conhecimento e o segundo como a Base de Fatos. A Máquina abstrata de Sistema de Informação (MSI) sendo a Máquina de Inferência.

O Projeto REMORA está em andamento e a MIND-AID é uma contribuição para ajudar no uso da metodologia no ambiente do banco, oferecendo facilidades de uso e de controle.

## 5.2. Avaliação da MIND-BFB:

Esta ferramenta automatizada, apesar de ser útil na prática, pode ser considerada como um protótipo, visto as inúmeras melhorias e aperfeicoamentos que poderão ser feitos mas que o tempo não permite.

Entretanto, ela é caracterizada pelo fato de que consegue integrar toda a especificação feita, faz uma verificação dessa especificação e gera os relatórios necessários. Ela cobre toda a fase de concepção e é bastante completa, tendo até um editor de texto e um "analisador" próprios. A BDD contendo os elementos da especificação e a linguagem formal associada a essa especificação estão numa forma próxima à do modelo REMORA.

O autor realisou dois testes com a ferramenta; um didático e simples (caso de reserva em hotel, mostrado neste documento) e outro um pouco maior (caso FINGER) cujos resultados encontram-se em [YAY1 88].

Outro funcionário do banco realisou um teste com a ferramenta considerando parte do sistema de câmbio; os resultados e uma impressão favorável encontram-se em [MAL 88].

A grande limitação desse CASE é a limitação no tamanho da BDD da especificação (A BDD do Turbo Prolog fica inteiramente na Memória Central, durante o uso) que pode comprometer o desempenho de alguns processos. Esse problema está para ser resolvido já que acaba de sair a nova versão do TURBO-PROLOG, que vai permitir manter a BDD na memória auxiliar (disco).

Antes de tudo o microcomputador precisa dispôr de pelo menos trezentos K bytes na memoria RAM para poder rodar os programas. Os diferentes módulos ocupam até duzentos K bytes; daí sobra só cem K bytes para a Base de Dados da especificação e para as diferentes pilhas que são montadas durante a execução. Por isso módulos como MC1 e MC3 não poderão ser rodados com Bases de Dados de mais de quarenta K bytes e, mesmo assim, depende da tarefa: Processos como normalização, definição de eventos ou ações não suportam que tenha muitos atributos não normalizados, objetos ou eventos de vez. É bom observar que a Base de Dados do caso de reserva em hotel ocupa cinco K bytes.

É importante saber que este sistema pode evoluir facilmente. A qualquer momento será possível acrescentar com muita facilidade novos módulos para realizar novas funcões ou modificar as antigas. Isso se deve às características da linguagem PROLOG.

## 5.3. Desenvolvimentos Adicionais:

A ferramenta MIND-AID poderá ser melhorada e estendida nas seguintes direções:

. Criar um módulo de Consulta da BDD da especificação que ofereça funções diversas, de sintaxes simples, para consultas rápidas e geração de relatórios (já essa versão do Prolog não oferece essas facilidades gerais de consultas, como é o caso para outras versões como ARITY e SIMPLE).

. Criar módulos que assistam o nível lógico da especificação a partir da especificação conceitual gerada com essa versão da MIND-AID.

. A MIND-AID da suporte à especificação textual da REMORA e gera um Diagrama da Dinâmica linear (o que é menos fácil de ler do que o diagrama padrão da metodologia REMORA). Então, seria bom criar módulos capazes de dar suporte à especificação gráfica em

sincronismo com a especificação textual (da MIND-AID), para que qualquer modificação numa das formas de especificação seja refletida na outra.

# B I B L I O G R A F I A:


[BAC 88] BACHMAN, C. - "A case for Reverse Engineering". Datamation, July, 1988, Vol. 34, No 13, p. 49 a 59, USA.

[BAR1 87] BARALDI, A. - "Metodologia". Publicação interna BFB, RID 0550F, 1987, BRASIL.

[BAR2 87] BARALDI, A. - "Macrometodologia de desenvolvimento de sistemas. Proposta inicial de técnicas de construção", publicação interna, BFB RID 0850F, 1987, BRASIL.

[BER 86] BERGLAND, G., D.; ZAVE, P. - "Special Issue on Software Design Methods", IEEE Transactions on Software Engineering, Vol se-12, No2, February, 1986, p.185 a 195, USA.

[BOR 86] "Prolog: the natural language of Artificial Intelligence", Borland International Editor, California, 1986, 224 p., USA.

[BOR 87] "Prolog Toolbox", Borland International Editor, California, 1987, 346 p., USA.

[BOU 87] BOUHOT, J.-P. - "Analyse des méthodes", L'informatique professionnelle, BOUHOT & LE GENDRE PUBLICATIONS, Mars 1987, P. 3 a 12, FRANÇA.

[CAR 86] CAMERON, J. - "An overview of JSD", IEEE Transactions on software engineering, Vol SE-12, No2, February 1986, p. 222 a 240, USA.

[CAM 87] CAMPOS, A., "Universidades na liderança da discussão do software", DATANEWS, 7 dezembro 1987, BRASIL.

[COD 70] CODD, E., F. - "A relational model of data for large shared data banks", Communications of ACM, vol SE-12, No2, February, 1970, USA.

[COH 88] COHEN, J. - "A view of the origins and development of Prolog", Communications of ACM, vol 31, No1, January, 1988, p.26 a 37, USA.

[DAR 87] DART, S., A.; ELLISON, R., J.; FEILLER, P., H.; HABERMAN, A., N. - "Software Development Environment", COMPUTER, Vol 20, No11, November, 1987, p. 18 a 28, USA.

[DEL 73] DELOBEL, C. - "Contribuitions théoriques à la conception et à l'évaluation d'un systeme d'informations appliqué à la gestion", these d'Etat, Université de Grenoble, 1973, FRANÇA.

[DEM 85] DEMOLOMBE, R. - "Bases de données, déjá trois générations", 01 informatique, no 186, JANVIER-FEVRIER 1985, p. 37 a 41, USA.

[FOU 82] FOUCAUT, O. - "Méthode et outil pour la conception des systemes d'information dans les organisations: Projet REMORA" These de doctorat es-sciences mathématiques, université de NANCY, 1982, 339 p., FRANÇA.

[GAN 83] GANE, C. & SARSON, T. - "Análise Estruturada de Ssistema", LTC Editora, 1983, 257 p., Rio de Janeiro, BRASIL.

[GRA 86] GRADY, B. - "Object-Oriented Development", IEEE Transactions on Software Engineering, Vol. se-12, No2, February, 1986, p. 211 a 221, USA.

[KOW 79] KOWALSKI, R. - "Logic For Problem Solving", North-Holland, Amsterdam, 1979, 287p., HOLLANDA.

[LEI 80] LEIFERT, S. - "Un langage de spécification des systemes d'information: un outil pour leur gestion" These de doctorat díngénieur , université de NANCY, 1980, 214 p., FRANÇA.

[LUC 87] LUCENA, C. - "Inteligência Artificial e Engenharia de Software", Publicações Academico-Cientificas, PUC-RJ/IBM, Jorge Zahar Editor, 1987, 305 p., BRASIL.

[MAC 86] MACDONALD, I., G. - "Information Engineering: Animportant, Automatable Methodology for the Design of Data Sharing Systems", IFIP 86, The Netherlands, 5-7 May, 1986, p. 173 a 223, HOLLANDA.

[McC 88] McCLURE, C. - "The case for Structured Development", PC TECH JOURNAL, Vol 6, No8, August, 1988, p. 50 a 67, USA.

[MAL 88] MALME, M., A., F., G. - "A Fábrica de Software", Dissertação de Formatura, EPUSP, DEP , São Paulo, 1988, 228 p., BRSIL .

[MAS 85] MASSAS, E.; LOTT, M. - "ADA", La Recherche, Vol 16, No 164, Mars, 1985, p. 350 a 358, FRANÇA.

[MEL 88] MELLO, R., C. - "Bancos de Dados não convencionais: a tecnologia do Banco de Dados e suas novas áreas de aplicação", VI Escola de Computação, Campinas, 1988, 730 p., BRASIL.

[MIC 88] MICHAEL., L., G. - "A Guide To Selecting Cases Tools". Datamation, July, 1988, Vol. 34, No 13, p. 65 a 66, USA.

[PRO 88] PROIX, C.; ROLLAND, C. - "A Knowledge Base for Information System Design", Univerité PARIS I & Laboratoires MASI, Centre de Calcul 12, Panthéon 75231 Paris, 1988, FRANÇA.

[RIN 88] RINGWOOD, G., A. - "Prolog88 and the dining logicians", Communications of ACM, Vol. 31, No1, January, 1988, p. 10 a 25, USA.

[ROL 79] ROLLAND, C. - "La conception des systemes de bureau", Procedings of AFCET congress, GRENOBLE, 1979, p. 59 a 77, FRANÇA.

[ROL 80] ROLLAND, C.; BENCI, G.; LEIFERT, S. - "Une méthode de conception canonique des bases de données", Anais do X Congresso Nacional de Informatica, São Paulo, SUCESSU, 1980, p. 41 a 53, BRASIL.

[ROL 82] ROLLAND, C.; RICHARD, C. - "The REMORA methodologie", in olle et alii (edit.) - Information Systemes Design Methodologies, a comparative review - North Holland IFIP 82, 1982, p. 369 a 426, HOLLANDA.

[ROL 83] ROLLAND, C.; RICHARD, C.;LEIFERT, S.- "A proposal for informations systems designs and management", CRIN - NANCY, université de NANCY, rapport interne, 1983, p. 83 a 109, FRANÇA.

[ROL 88] ROLLAND, C.; FOUCAUT, O.; BENCI, G. - "Conception des Systemes d'Informationn: La méthode REMORA", EYROLLES (edit), Paris, 1988, 351 p., FRANÇA.

[RUS 86] RUSSEL, J. - "Action Diagrams for systems analysis", Data Processing, Vol 28, No4, May 1986, p. 179 a 182, USA.

[THI 85] THIERRY, O. - "LASSIF: un langage de spécification des systemes d'information: un logiciel d'aide à la spécification des systemes d'information", These de doctorat es-sciences mathématiques, université NANCY, 1985, 390 p., FRANÇA.

[UCG 88] United Communications Group - "Improving Large Applications on Software Development", I/S Analyzer, March, 1988, p. 1 a 16, USA.

[WAN 80] WARNIER, D.- "Logique de Construction de Programme", Editions d'Organisation, Paris, 1980, FRANÇA.

[WAR 86] WARD, P., T. - " The Transformation Schema : an extension of Data Flow Diagram to represent control and timing", IEEE Transactions on Software Engineering, Vol SE-12, No2, February. 1986, p. 198 a 210, USA.

[YAU 86] YAU, S.; EFFERY, J.; TSAI, J-P. - "A survey of software design techniques", IEEE Transactions on Software, Vol SE-12, No6, June, 1986, p. 713 a 721, USA.

[YAY1 88] YAYA, T. - "MIND-AID: Guia do usuário", Publicação Interna, BFB, YAY005R, 1988, 22 p., BRASIL.

[YAY2 88] YAYA, T.; MALME, M., A., F., G. - "Estudo de CASEs",

Publicação Interna, BFB, YAY005G, 1988, 28 p., BRASIL.

[YOU 86] YOURDON, E. - "What ever happened to structured analysis ?", DATAMATION, Vol 32, No11, p. 133 a 138, 1986, USA.

# A N E X O 1:

## G L O S S Á R I O:

**1FN:** Primeira Forma Normal

**2FN:** Segunda Forma Normal

**3FN:** Terceira Forma Normal

**3FNP:** Terceira Forma Normal Permanente

**ACi:** ação (ou operação) i

**BDD:** Base de Dados

**BFB:** Banco Francês e Brasileiro

**CASE:** Computer-Aided Software Engineering (ferramenta de apoio à concepção)

**DCDC:** Dependência Cronologica Direta Condicional

**DCDS:** Dependência Cronologica Direta Sistematica

**DF:** Dependência Funcional

**DFP:** Dependência Funcional Permanente

**EC:** Esquema Conceitual

**ES:** Engenharia de Software

**ESM:** Esquema de Sincronização de Modulos

**EVi:** Evento i

**IA:** Inteligência Artificial

**ISDEL:** Information System DEfinition Language, linguagem de modelagem da REMORA, [LEI 80]

**MER:** Modelo Entidade-Relacionamento

**MIND-AID:** ferramenta proposta nesse trabalho para apoio à concepção usando a REMORA

**MIND-BFB:** Macrometodologia INtegrada de Desenvolvimento de sistemas do Banco Francês e Brasileiro; mas, tambem, nome da linguagem de especificação e do ambiente de edição e de compilação propostos nesse trabalho.

**MSI:** Maquina de Sistema de Informação, descrição teórica de uma máquina abstrata para o gerênciamento do SI, [FOU 82].

**OBi:** objeto i

**OICSI:** Outil Interactif pour la Conception de Systemes d'Information, Sistema Especialista para a modelagem conceitual atraves a REMORA, em andamento na FRANCA, [PRO 88].

**REMORA:** nome da metodologia de especificação de sistema proposta pelo grupo de pesquisa de COLETTE ROLLAND, franca.

**SGBD:** Sistema de Gestão de Base de Dados

**SI:** Sistema de Informação

# A N E X O  2:

## Sintaxe da linguagem MIND-BFB:

A notação BNF seguinte é usada para definir a sintaxe da linguagem de especificação conceitual MIND-BFB:

<termo> : elemento não-terminal

{ X }* : representa uma ou mais repetição de X

[ X ] : significa que X é opcional

X | Y : indica que tanto X como Y podem ser usados, mas não simultaneamente.

As cadeias de caracteres sublinhadas representam os simbolos predefinidos na linguagem.

```
<programa> ::= { <clausula}*

<clausula>     ::= <fato>. |
                   <regra>.

<fato>         ::= <exp-rel>

<regra>        ::= <exp-rel> se <cond>

<cond>         ::= <lst-alvos> [ ou <lst-alvos> ]

<lst-alvos>    ::= <alvo> |
                   <alvo> e <lst-alvos>

<alvo>         ::= <exp-rel> |
                   <comprs> |
                   <manip-BDD> |
                   <monte-liste> |
                   <fonc> ( <lst-termos> ) |
                   nao <exp-rel>  |
                   fcsgn

<exp-rel>      ::= <nom> ( <lst-termos> )

<montelst>     ::= monteliste ( <var>, <exp-rel>, <var> )

<manip-BDD>    ::= incluir <exp-rel> |
                   excluir <exp-rel>
```

```
<lst-termos> ::= <termo> |
                 <termo>, <lst-termos>

<termo>      ::= <numero> |
                 <lista> |
                 <cadeia> |
                 <var>

<lista>      ::=  [] |
                  [ <elm-lst> ]

<elm-lst>    ::= <termo> [ : <termo> ] |
                 <termo>, <elm-lst>

<compar>     ::= <exp> <comp> <aritm> |
                 <aritm> <comp> <aritm>

<comp>       ::= = | <= | >= | >< | <> | =< | => | < | >

<ascii>      ::= <const> |
                 <cadeia> |
                 <var>

<aritm>      ::= <exp> <soma> <aritm> |
                 <exp>

<exp>        ::= <fator> <multiplie> <aritm> |
                 <fator>

<fator>    ::= <var> |
                 <numero> |
                 <aritm> |
                 <func> ( <aritm> )

<soma>       ::= + | -

<multiplie> ::= * | / | div | mod

<func>   ::= abs | max | min | sort | total

<var>        ::= ( <majusc> | _ ) [ <nome> ]

<const>      ::= <minusc> [ <nome> ]
```

## A N E X O 3:

## LISTAGEM COMENTADA DO MÓDULO MC1:

```
/* ----------------------------------------------------------------*/
/*
                MC1.PRO
Esse modulo permite criar  o esquema estatico : tipos, atributos,
normalização das relações e c_relacionamentos.                    */

/* ----------------------------------------------------------------*/
code=4000
nowarnings
DOMAINS
  ROW, COL, LEN, ATTR    = INTEGER
  STRINGLIST = STRING*
  INTEGERLIST = INTEGER*
  KEY     = cr; esc; break; tab; btab; del; bdel; ctrlbdel; ins;
        end ; home ; fkey(INTEGER) ; up ; down ; left ; right ;
        ctrlleft; ctrlright; ctrlend; ctrlhome; pgup; pgdn;
        ctrlpgup; ctrlpgdn; char(CHAR) ; otherspec

/* -------------------------------------------------------*/

                    DOMAINS
/* -------------------------------------------------------*/


  FNAME=SYMBOL
  TYPE = int(); str(); real(); tipo1(); tipo2(); atr2(); valid(); estr()

     stringlist2 = stringlist*
     lista_atr = atr*
     atr = atributo(string,string,string,string)
/* -------------------------------------------------------*/

                    DATABASE
/* -------------------------------------------------------*/

  insmode
  actfield(FNAME)
  screen(SYMBOL,DBASEDOM)
  value(FNAME,STRING)
  field(FNAME,TYPE,ROW,COL,LEN)
  txtfield(ROW,COL,LEN,STRING)
  windowsize(ROW,COL).
  notopline

  windowstart(ROW,COL)
  mycursord(ROW,COL)

  lineinpstate(STRING,COL)
  lineinpflag
```

```
/* ---------------declaracao da Base de Dados --------------*/

      c_objeto(string,string,stringlist,real,real)
      c_relacionamento(string,string,stringlist,string,real,real)
      tipo(string,string,stringlist,string,string)
      atributo(string,string,string,string)
      atributon(string,string,string,string)
      df(stringlist,string)
      chave(string,stringlist)


/* ----------------------------------------------------------*/

                PREDICATES
/* ----------------------------------------------------------*/

valable(FNAME)
limpa_value

inicio(string)
debut(integer,string)

tipos
objetos
normalizacao
corp_MC14(stringlist2)
relacionamentos

calendrier(real)
membro(string,stringlist)
membrol(stringlist,stringlist2)
pergunta(stringlist,stringlist)
insere_df(stringlist,string)
junta(stringlist,stringlist,stringlist)
juntal(stringlist2,stringlist2,stringlist2)
dar_nome(stringlist2)
unificar(stringlist2)
rassembler(stringlist,string)
verifica(stringlist2,stringlist2)
eliminer(stringlist2,stringlist2)
intersect(stringlist,stringlist,stringlist)
lint_listAttr(integerlist,stringlist,stringlist)
position(string,stringlist,integer)
list_str(stringlist,string)
str_list(string,stringlist)
atributon_atributo

deletesc
deletedf
deletedb

inic(string)
traiter(stringlist2,stringlist2)
relac(stringlist,stringlist2)
```

```
deux(stringlist,stringlist2,integer,integer)
declarer(stringlist)
subconjunto(stringlist,stringlist)

include "tpreds.pro"
include "menu.pro"

PREDICATES
  makestatus(ATTR,STRING)
  nondeterm tempstatus(ATTR,STRING)
  removestatus
  refreshstatus
  changestatus(STRING)

CLAUSES
  makestatus(ATTR,TXT):-
     shiftwindow(OLD),
     makewindow(83,ATTR,0,"",24,0,1,80),
     field_str(0,0,80,TXT),
     shiftwindow(OLD).

  tempstatus(ATTR,TXT):-makestatus(ATTR,TXT).
  tempstatus(_,_):-removestatus,fail.

  removestatus:-
     shiftwindow(OLD),
     shiftwindow(83), removewindow,
     shiftwindow(OLD).

  refreshstatus:-
     shiftwindow(OLD),
     shiftwindow(83),
     shiftwindow(OLD).

  changestatus(TXT):-
     shiftwindow(OLD),
     shiftwindow(83),
     field_str(0,0,80,TXT),
     shiftwindow(OLD).

include "lineinp.pro"
include "scrhnd1.pro"
include "boxmenu.pro"

include "filename.pro"

/* ------------------ inicio do modulo --------------------------*/
                    GOAL

system(cls),
deletesc,
deletedb,
makewindow(11,7,7,"definicão do sub-esquema estatico",0,0,3,80),
makestatus(7," De o nome que deseja para o sub-esquema estatico"),
```

```
readfilename(10,10,7,7,"BFB","ESQEST.BFB",Arq),
changestatus(""),
 consult("mc1.scr"),
inicio(Arq).

/* ----------------------------------------------------------
                CLAUSES


/* ---------------- leitura do nome do arquivo ---------

inicio(Arq) :-

 existfile(Arq), !, consult(Arq), inic(Arq).

inicio(Arq) :-
          inic(Arq).

inic(Arq) :-
menu(10,10,7,7,["definicão TIPOS",
                "entrada ATRIBUTOS",
                "NORMALIZACÃO ==> CRIACÃO C_OBJETOS",
                "decidindo  C_RELACIONAMENTOS"],
                "escolha a tarefa desejada",1,Rep),
debut(Rep,Arq).


debut(1,Arq) :- !, tipos, inic(Arq).
debut(2,Arq) :- !, objetos, inic(Arq).
debut(3,Arq) :- !, normalizacao, inic(Arq).
debut(4,Arq) :- !, relacionamentos, inic(Arq).

debut(_,Arq) :-     deletesc,
               save(Arq),
               deletedb.
/* ----------------- limpar as telas ----------------- */

deletesc :-    retract(screen(_,_)),fail.
deletesc :-    retract(value(_,_)),fail.
deletesc :-    retract(field(_,_,_,_,_)),fail.
deletesc :-    retract(txtfield(_,_,_,_)),fail.
deletesc :-    retract(windowsize(_,_)),fail.
deletesc :-    retract(actfield(_)),fail.
deletesc :-    retract(df([],_)), fail.
deletesc :-     retract(insmode), fail.
deletesc.
/* ----------------- limpar as occorencias da BDD -------

deletedb :-retract(c_objeto(_,_,_,_,_)), fail.
deletedb :-retract(c_relacionamento(_,_,_,_,_,_)), fail.
deletedb :-retract(tipo(_,_,_,_,_)), fail.
deletedb :-retract(atributo(_,_,_,_)), fail.
deletedb :-retract(df(_,_)), fail.
```

```
deletedb :-retract(chave(_,_)), fail.
deletedb.

/* ----------------- partes principais ----------------- */

/*%%%%%%%%%*/            tipos :-

          shiftscreen(mc11),
          createwindow(off),
          scrhnd(off,KEY11), not(KEY11=esc),

value("NomeTipo",NomeTipo), value("Estr",NomeEstr),
value("Valid",Valid),       value("Libelo",Libelo),
value("Valor",Valor),

          assertz(tipo(NomeTipo,NomeEstr,[Valor],Valid,Libelo)),
          removewindow, limpa_value, !.

tipos :- removewindow, limpa_value.

/* --------- entrada dos atributos ----------- */
/*%%%%%%%%%*/            objetos :-

           shiftscreen(mc12),
           createwindow(off),
          scrhnd(off,KEY12), not(KEY12=esc),

value("NomeTipo",NomeTipo), value("Atr",Atr),
value("Valid",Valid),       value("Libelo",Libelo),

              assertz(atributon(Atr,NomeTipo,Valid,Libelo)),
        removewindow, limpa_value, !.

objetos :- removewindow, limpa_value.

/* --------- normalizacao para criacao dos objetos ----------- */
/*%%%%%%%%%*/          normalizacão  :-

                     deletedf,

     findall(Atr,atributon(Atr,_,_,_),LA),
     atributon_atributo,
          pergunta(LA,LA),

                corp_MC14(_).


pergunta([],_) :- !.
pergunta([Tete|Reste],LA) :-
changestatus(" QUAIS SÃO OS ATRIBUTOS QUE CARACTERIZAM ESSE ATRIBUTO ?"),
                boxmenu_mult(5,15,15,40,7,7,LA,Tete,[1],Choix),
                 lint_listAttr(Choix,Cle,LA),
                 insere_df(Cle,Tete),
                 pergunta(Reste,LA).
```

```
lint_listAttr([],[],_).
lint_listAttr([T|Q],[X|Y],LA) :- position(X,LA,T),
          lint_listAttr(Q,Y,LA).


position(X,[X|Y],1):- !.
position(X,[_|Y],N) :- N1 = N - 1, position(X,Y,N1).

insere_df(Cle,Tete) :- df(Cle,Tete),!.
insere_df(Cle,Tete) :- assertz(df(Cle,Tete)).

/*%%%%%%%%*/            corp_MC14(Lcle) :-

      findall(Cle,df(Cle,Atr),Lcle1),
          verifica(Lcle1,Lcle).

unificar([T|Q]) :- df(T,Atr),!, retract(df(T,Atr)),
              rassembler(T,Atr),
              unificar(Q).
unificar([]).

rassembler(T,Atr) :- df(T,Atr2),!, retract(df(T,Atr2)),
                concat(Atr,"  ",A), concat(A,Atr2,Atr3),
                rassembler(T,Atr3).

rassembler(T,Atr) :- assertz(df(T,Atr)).

dar_nome([T|Q]) :- !, df(T,Atr), junta(T,[Atr],Ri),

          limpa_value,

          changestatus("" ),
          calendrier(Datee),

                list_str(Ri,Rii),

          assertz(value("Ri",Rii)),

          shiftscreen(mc14),
          createwindow(off),
          scrhnd(off,_),

          value("Codigo",Codigo),
          value("Nome",Nome),

          assertz(chave(Nome,T)),
          assertz(c_objeto(Codigo,Nome,Ri,Datee,0)),
          limpa_value, removewindow,

          dar_nome(Q).

dar_nome([]) :- removewindow.
```

```
verifica([[]|Q],Lcle) :- !, verifica(Q,Lcle).
verifica([T|Q],Lcle)  :- !, juntal(Lcle,[T],L), verifica(Q,L).
verifica([],Lcle) :- unificar(Lcle),!,
                eliminer(Lcle,LL).

/* ---------------- pesquiza de relacionamentos --------------- */
/*%%%%%%%%*/            relacionamentos :-


     findall(T,chave(_,T),Lchaves),
     traiter(Lchaves,Lchaves).

traiter([],_) :- !.
traiter([C1|R],Lchaves) :- relac(C1,Lchaves), !,
                       traiter(R,Lchaves).
traiter([_|R],Lchaves) :- traiter(R,Lchaves).


relac(C1,Lchaves) :- deux(C1,Lchaves,0,_),
                declarer(C1).

deux(_,_,2,_) :- !.
deux(C1,[C2|R],N1,N2) :-      subconjunto(C2,C1), !,
                     N2 = N1 + 1,
                     deux(C1,R,N2,N).

deux(C1,[_|R],N1,N2) :- deux(C1,R,N1,N2).


declarer(C1) :- chave(Nomeobj,C1),
          c_objeto(Codigo,Nomeobj,RI,Date,0),
          list_str(RI,RII),
          assertz(value("RI",RII)),
           shiftscreen(mc15),
           createwindow(off),
changestatus
("de o nome da regra de relacionamento e F10, para criar / ESC, se nao"),

          scrhnd(off,KEY15), not(KEY15=esc),

value("Nome",Regra),
retract(c_objeto(Codigo,Nomeobj,RI,Date,0)),
assertz(c_relacionamento(Codigo,Nomeobj,RI,Regra,Date,0)),

          removewindow, limpa_value, !.

declarer(_) :- removewindow, limpa_value.


/* ---------------- algumas facilidades --------------- */


membro(Tete,[Tete|Reste]) :- !.
membro(X,[_|Reste]) :- membro(X,Reste).
```

```
membroI(Tete,[Tete|_]) :- !.
membroI(X,[_|Reste]) :- membroI(X,Reste).

calendrier(Datee) :- date(A,M,D),
                B = A - 1900,
                bound(B),
                str_int(B1,B),str_int(M1,M),str_int(N1,D),
                str_len(M1,1), !,
                concat("0",M1,M2),
                concat(B1,M2,Dateee),concat(Dateee,N1,Dat),
                str_real(Dat,Datee).

calendrier(Datee) :- date(A,M,D),
                B = A - 1900,
                str_int(B1,B),str_int(M1,M),str_int(N1,D),
                concat(B1,M1,Dateee),concat(Dateee,N1,Dat),
                str_real(Dat,Datee).


junta([],Liste,Liste)    :-!.
junta([X|L1],L2,[X|L3]) :- junta(L1,L2,L3).

juntaI([],Liste,Liste)    :-!.
juntaI([X|L1],L2,[X|L3]) :- juntaI(L1,L2,L3).

eliminer([],LM) :-    dar_nome(LM).


eliminer([T|Q],LM) :- not(membroI(T,Q)),!, juntaI(LM,[T],LMT),
                eliminer(Q,LMT).

eliminer([T|Q],LM) :- eliminer(Q,LM).

intersect([],[],LA) :- !.

intersect([X|Y],[X|L1],L2) :- membro(X,L2),
                     intersect(Y,L1,L2),!.

intersect(Y,[_|L1],L2) :- intersect(Y,L1,L2).


limpa_value :- retract(value(_,_)),fail.
limpa_value.

list_str([],"") :- !.
list_str([T|Q],S) :-  list_str(Q,S1), concat(T,"  ",T1), concat(T1,S1,S).

str_list(S,[T|Q]) :-
     fronttoken(S,T,S1), !,
     str_list(S,Q).
str_list(_,[]).
```

```
subconjunto([],_) :-!.
subconjunto([T|Q],L) :- membro(T,L),
                subconjunto(Q,L).

atributon_atributo :-
          atributon(A,B,C,D),
          assertz(atributo(A,B,C,D)),
          retract(atributon(A,B,C,D)), fail.
atributon_atributo.

deletedf :- retract(df(_,_)), fail.
deletedf.


/* ------------------------------------------------------------ */

  noinput("RI"):- !.
  noinput(_) :- fail.

  field_action(_):-fail.

/* ------------------------------------------------------------ */

  field_value(FNAME,VAL):-value(FNAME,VAL),!.
```

```
tipo("TIPOQ","simples",["A10"],"valores(\"SIMPLES\",\"MEDIO\",\"LUXO\")","nil")
tipo("TIPOR","simples",["A10"],"valores(\"SIMPLES\",\"MEDIO\",\"LUXO\",\"CANCEL\")","nil")
tipo("NOME","simples",["A30"],"nil","nil")
atributo("ESTADOQ","CADEIA","valores(\"Livre\",\"Ocupado\")","nil")
atributo("DATAMUDQ","DATA","nil","nil")
atributo("TIPOQ","TIPOQ","nil","nil")
atributo("PRECOQ","INTEIRO","nil","nil")
atributo("DURACAO","INTEIRO","nil","nil")
atributo("CUSTO","INTEIRO","regra(CUSTO)","nil")
atributo("NOQ","NOQ","tabela(QuartDisp)","nil")
atributo("ESTADOR","CADEIA","valores(\"FEITA\",\"CANCELADA\")","nil")
atributo("DATAMUDR","DATA","nil","nil")
atributo("INICIOR","DATA","nil","nil")
atributo("DATAFIM","DATA","nil","nil")
atributo("DATAINICIO","DATA","nil","nil")
atributo("NOCLI","NOCLI","nil","nil")
atributo("NOMECLI","NOME","nil","nil")
atributo("ENDERCLI","NOME","nil","nil")
atributo("TIPOR","TIPOR","nil","nil")
atributo("DIA","DATA","nil","nil")
chave("ESTADOQ",["NOQ,ESTADOQ"])
chave("QUARTO",["NOQ"])
chave("RESERVA",["INICIOR,NOCLI,NOQ"])
chave("ESTADOR",["INICIOR,NOCLI"])
chave("CLIENTE",["NOCLI"])
chave("PEDIDO",["DATAINICIO,NOCLI"])
chave("CALENDARIO",["DIA"])
```

" Assim, recomeçando uma obra vinte vezes, ao escrever quatro palavras, apagarei três..."

BOILEAU