UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE DIPLOMA

ANÁLISE COMPARATIVA DE ALGUMAS TÉCNICAS UTILIZADAS NA ANÁLISE DE SISTEMAS

O SUPERVISOR

dr. JOÃO MORENO

ELABORADO POR:

ANITA ALBINO GUAMBE

CAROLINA JOSÉ NHAMBIRE

MAPUTO, MAIO DE 1993

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE MATEMÁTICA E INFORMÁTICA

TRABALHO DE DIPLOMA

ANÁLISE COMPARATIVA DE ALGUMAS TÉCNICAS UTILIZADAS NA ANÁLISE DE SISTEMAS

ELABORADO POR:

ANITA ALBINO GUAMBE e

CAROLINA JOSÉ NHAMBIRE

Maputo, Maio de 1993

DECLARAÇÃO

Declaramos que este trabalho é resultado da nossa própria investigação, que não foi submetido para outro grau que não seja o indicado - Licenciatura em Informática pela Universidade Eduardo Mondlane

Anita Albino Guambe

Carolina José Nhambire

Maputo, Maio de 1993

AGRADECIMENTOS

Na preparação deste trabalho, contámos com o grande apoio do nosso supervisor, o dr. João Moreno, docente do departamento, que com a sua inteligência e dedicação soube-nos orientar, pelo que expressamos o nosso maior agradecimento.

Queremos também agradecer a solidariedade demonstrada por todos os professores e colegas no dia a dia, durante os anos do curso. Igualmente endereçamos a nossa gratidão a todos os colegas pelo apoio prestado na realização deste trabalho.

Finalmente, expressamos o nosso agradecimento a familiares e amigos que ao longo do curso puderam encorajar-nos.

RESUMO

Pretende-se com este trabalho fazer um estudo comparativo de quatro técnicas usadas na análise de processos, nomeadamente, Diagramas de Fluxos de Dados (DFD's), Hierarchy/Input/Process/Output (HIPO), Fluxogramas e Pseudocódigo.
Este estudo visa fundamentalmente a análise aprofundada destas técnicas, através de exemplos ilustrativos, permitindo deste modo, encontrar os pontos comuns, diferenças e, consequentemente, as vantagens e desvantagens de cada técnica.

Finalmente, apresentam-se as conclusões gerais que resultaram do processo de estudo.

LISTA DAS FIGURAS

ÍNDICE

1 - INTRODUÇÃO

É objectivo deste trabalho aprofundar o estudo de algumas técnicas usadas em análise de sistemas, mais concretamente na análise de processos, usando essencialmente metodologias estruturadas.

Quatro técnicas, nomeadamente, Diagramas de Fluxos de Dados (DFD's), Hierarchy/Input/Process/Output (HIPO), Fluxogramas e Pseudocódigo, são aqui desenvolvidas de forma a dar uma maior visão sobre o uso destas ferramentas e as particularidades de aplicação desenvolvidas pelas diferentes metodologias.

O trabalho visa fundamentalmente ampliar o conhecimento técnico-científico destas ferramentas e permitir uma rápida reflexão, tendo em conta as vantagens e desvantagens, na escolha de metodologias a usar para a resolução de problemas reais futuros.

Em vista da semelhança quanto aos objectivos de cada técnica, adoptou-se como metodologia, uma abordagem comparativa das mesmas. Assim, analisaram-se as vantagens e desvantagens do DFD relativamente ao HIPO, tendo-se utilizado o mesmo procedimento no que se refere a Fluxogramas e Pseudocódigo.

Os exemplos ilustrativos usados ao longo do trabalho são do trabalho preparatório, realizado pelas autoras no Banco de Moçambique [Nhambire e Guambe, 1993], no qual desenvolvem um sistema usando fundamentalmente metodologias estruturadas de análise de sistemas.

No âmbito da elaboração deste trabalho, por razões de comodidade da análise, o estudo dos DFD's e do HIPO esteve a cargo de Carolina e por sua vez a abordagem sobre Fluxogramas e Pseudocódigo foi feita por Anita. Assinale-

se, no entanto, que a respeito desta divisão de tarefas, a obra final é fruto de um trabalho de equipa e, por conseguinte, cada uma das autoras está perfeitamente familiarizada com cada uma das ferramentas consideradas.

2 - DIAGRAMA DE FLUXO DE DADOS (DFD's)

Esta técnica é usada por várias metodologias estruturadas, facto que demonstra a sua potencialidade.

É obvio, no entanto, que cada metodologia use esta técnica com definições e simbologias específicas, demonstre as suas vantagens e desvantagens, etc.

A seguir apresentam-se as diferentes versões do uso desta ferramenta.

2.1 - DEFINIÇÕES

Consultada a bibliografia diversa que referencia a técnica dos DFD's, e analisando as diferentes formas de acepção desta técnica, compilou-se o que se segue:

[Ashworth e Goodland, 1990], é de opinião que um DFD's é uma representação diagramática do fluxo de informações dentro de um sistema, mostrando a sua entrada e saída, as transformações dos dados e onde a informação é armazenada.

[Barker, 1990], define DFD como sendo a técnica usada para mostrar o fluxo de informação entre diferentes processos.

[Davis, 1987], diz que o DFD é uma excelente ferramenta de comunicação entre o analista e o utente, pois trata-se de uma representação gráfica do sistema lógico, independente do hardware, software, da organização de ficheiros ou da estrutura de dados, de fácil compreensão pelos utentes não técnicos. Por isso mesmo considera-o como um bom ponto de partida para o projecto de sistemas.

[DeMarco, 1978], define o DFD como sendo uma representação

em rede de um sistema, ele retrata o sistema partindo de componentes, incluíndo os interfaces entre os componentes indicados.

[Meilir, 1988], define o DFD como sendo uma representação em rede de um sistema que mostra as componentes activas do sistema e os interfaces de dados entre eles. È também, informalmente, conhecido como gráfico de bolha por ser constituído por pequenos círculos.

Para [Neto et al, 1988], a técnica dos DFD's é uma ferramenta descendente "top-down",uma vez que parte do nível mais alto de abstracção para o nível de maior detalhe. A sua representação gráfica é uma técnica formal de modelagem de processos que permite compartilhar o modelo entre a comunidade dos analistas e usuários.

[Sommerville, 1992], por sua vez, afirma que a técnica dos DFD's é parte integrante de vários métodos de desenho e ferramentas CASE (Computer Aided Software Engineering).

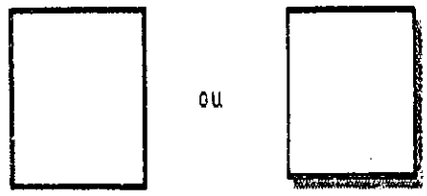
É fácil notar que as definições têm a mesma essência, diferindo apenas na linguagem usada em cada metodologia.

Estas definições podem resumir-se no seguinte:

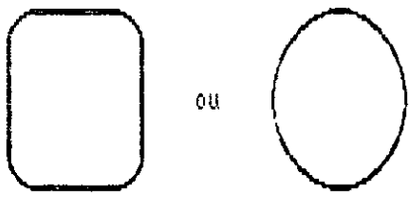
Um diagrama de fluxo de dados é uma técnica importante de análise de sistemas, ele é uma ferramenta "top-down", que serve de elo de comunicação entre o analista e o utente. A sua representação é gráfica para permitir dar uma visão da imagem total (mostrar todos os fluxos internos do sistema) e definir claramente as fronteiras do sistema em estudo.

2.2 - SIMBOLISMO

As metodologias que usam esta ferramenta, (geralmente as de análise estruturada), utilizam uma notação mais ou menos similar. Por exemplo, [Davis, 1987], usa a simbologia ilustrada na fig. 2.1.



Origem ou destino dos dados



Processo que transforma os dados

Depósito de dados

Fluxo de dados

FIG. 2.1 - SÍMBOLOS DO DFD, NOTAÇÃO DE [Davis, 1987]

[Neto et al, 1988], usa a mesma notação para o fluxo de dados e depósito de dados. Para o processo usa como simbolo um rectângulo com bordas arredondadas, e para representar o agente externo (que na primeira notação tem a designação de origem ou destino), conforme se ilustra na fig. 2.2.

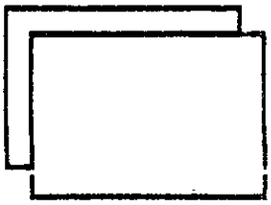


FIG. 2.2 - SÍMBOLOS DO DFD, NOTAÇÃO DE [Neto et al, 1988]

Para a metodologia SSADM (Structured Systems Analisys and Design Metodology), [Ashworth e Goodland, 1990], os símbolos idênticos aos já referenciados, são os do depósito de dados e fluxos de dados. A figura 2.3 ilustra os símbolos usados para representar o agente externo e o processo, respectivamente.

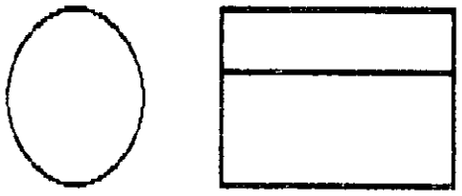


FIG. 2.3 - SÍMBOLOS DO DFD, NOTAÇÃO DE [Ashworth e Goodland, 1990]

[Meilir, 1988] na sua notação usa também a seta para representar o fluxo de dados, o círculo para o processo, o rectângulo para o agente externo e duas linhas paralelas horizontais para representar o depósito de dados, como a seguir se ilustra.

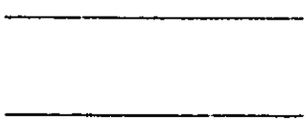


FIG. 2.4 - SÍMBOLOS DO DFD, NOTAÇÃO DE [Meilir, 1988]

[Sommerville, 1992] representa o depósito de dados por um quadrado, e o agente externo por um círculo, ademais, usa duas palavras chaves "AND" e "OR" para unir os fluxos de dados.

Estas palavras chaves são usadas em conjunção com um

símbolo em forma de arco. Os restantes símbolos são idênticos aos usados por [Neto et al, 1988].
[Barker, 1990] usa a simbologia idêntica a usada na metodologia SSADM,exceptuando a entidade externa, cujo símbolo é um rectângulo com bordas arredondadas.

[Gane e Sarson, 1979] usa notação muito similar a de [Davis, 1987] para o fluxo de dados e depósito de dados. Para representar o processo e a entidade externa,adoptou apenas um rectângulo com bordas arredondadas, (alegadamente por o círculo que também é usado por [Davis, 1988], dificultar a escrita legível) e o quadrado, respectivamente, conforme se mostra na fig. 2.5.

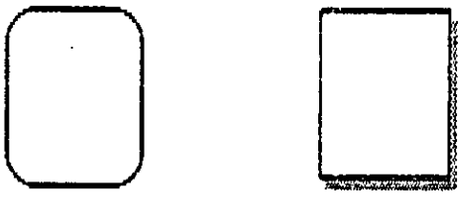


FIG. 2.5 - SÍMBOLOS DO DFD, NOTAÇÃO DE [Gane e Sarson, 1979]

Tal como [Davis, 1987], [DeMarco, 1978] usa para a entidade externa, processo e fluxo de dados, o quadrado, o círculo e a seta, respectivamente. Para o depósito de dados usa uma linha recta, fig 2.6.

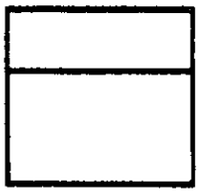
FIG. 2.6 - SÍMBOLOS DO DFD, NOTAÇÃO DE [DeMarco, 1978]

Faz também alusão ao uso de informações procedurais adicionais, compostas pelos símbolos (*), que significa "E" e (+) que siginifica "OU", retratando a conjunção e disjunção respectivamente, como acontece na notação usada por [Sommerville, 1992]. Contudo, desencoraja o seu uso de um modo geral, alegadamente por aumentar a redundância no DFD, uma vez que, as anotações estão contidas nas

miniespecificações, que fazem parte integrante, juntamente com os DFD's e Dicionário de dados, da especificação estruturada completa.

Partindo da simbologia aqui representada, achamos que seria mais fácil obter-se uma escrita legível num DFD se se adoptar uma notação idêntica a da SSADM mas com entidade externa representada por um rectângulo, como se usa na notação de [Neto et al, 1988], isto para permitir uma escrita legível em todas as componentes de um DFD, uma vez que, símbolos como círculo ou oval dificultam esse aspecto. Assim sendo, a simbologia proposta seria a seguinte:

Entidade externa



Processo

Depósito de dados

Fluxo de dados

FIG. 2.7 - SÍMBOLOS PROPOSTOS

2.3 - CONSTRUÇÃO DOS DFD's

[DeMarco, 1978] diz que "os DFD's são uma técnica importante de análise de sistemas", e são usados para:

1 - Definição de fronteiras .

Os diagramas mostram claramente as fronteiras do sistema representado.

2 - Revisão completa da análise

A construção do diagrama e sua comparação com outras técnicas principais do SSADM, permite ter a certeza de que todos os fluxos de informação, depósitos de dados e actividades dentro do sistema estão sendo considerados.

3 - Base para a especificação de programas

O DFD denota as áreas funcionais do sistema e portanto, o conjunto de programas necessários.

Para a construção de um DFD, é necessário primeiro estabelecer os principais mecanismos e regras de desenho.

2.3.1 - PRINCÍPIOS GERAIS DE DESENHO

O desenho de um DFD deve obedecer aos seguintes princípios, [DeMarco, 1978]:

1 - Atribuição de nomes significativos

Os nomes devem ser atribuídos em função dos fluxos de dados que se movimentam para dentro e para fora do diagrama.

2 - Controlo da semântica

Deve-se dar novos nomes a fluxos de dados modificados e evitarem-se falsos fluxos de dados. Por exemplo, se existir um processo "A" que acciona um processo "B" e este, por sua vez, acciona um processo "C", se existir em paralelo um fluxo de dados que saia do processo "A" para o processo "C", este fluxo é falso e, por isso, deve ser retirado do diagrama.

3 - Controlo da redundância

Devem ser identificados todos os fluxos de dados pseudónimos (fluxos com nomes diferentes mas com o mesmo conteúdo de dados). Deve-se evitar, também, a repetição de arquivos e processos.

4 - Manutenção da consistência entre níveis de detalhe

Os arquivos locais são visíveis a partir de um determinado nível e devem ser exclusivos de um processo. São desnecessários ao mais alto nível para a compreensão do sistema.

5 - Manutenção do equilíbrio da representação

Deve-se tentar de modo que em qualquer nível se obtenha um número semelhante de processos e que estes tenham um nível de complexidade similar. Num DFD devem ser representados, no máximo, 10 processos.

É de notar que, em conformidade com os princípios gerais, existem regras específicas para cada componente do DFD. As metodologias SSADM [Ashworth e Goodland, 1990] , Case*Method [Barker, 1990] e Análise Estruturada e Especificação do Sistema [DeMarco, 1978], são as que abordam estas regras de forma sintética.

2.3.2 - COMPONENTES DE UM DFD

1 - "**Entidade externa** é uma pessoa ou empresa que está fora do contexto do sistema, que é a fonte ou receptor de dados do sistema." [DeMarco, 1978, p.55]

Uma entidade externa deve ter, [Ashworth e Goodland, 1990] e [Barker, 1990].

- Um único nome
- Uma descrição concisa e significativa
- Todos os atributos completamente definidos
- Todas as relações bem definidas
- Sinónimos (se houver), que são únicos dentro do contexto.

2 - "**Processo** é a componente que transforma os fluxos de dados de entrada em fluxos de dados de saída" [DeMarco, 1978].

Um processo deve: [Ashworth e Goodland, 1990] e [Barker, 1990].

- Ter uma descrição concisa e significativa, que começa com um verbo
- Ser único no contexto
- Ter um conjunto de filhos que realizam todas as funções dos pais
- Referir-se apenas a entidades que estão na sua alçada
- Ter função lógica especificada.

3 - "**Depósito de dados** é onde se guardam temporariamente os dados, pode ser um arquivo, uma pasta de documentos, uma fita magnética, etc." [DeMarco, 1978].

Um depósito de dados deve: [Barker, 1990].

- Ter um nome (o usado no sistema actual se possível)
- Ter a capacidade de construção de seus fluxos de dados de entrada
- Ter a capacidade de agir como uma fonte para qualquer fluxo de dados de saída
- Ser constituída de atributos de entidades conhecidas ou, elemento de dado cuja fonte ou destino é clara.

4 - "**Fluxo de dados** é um tubo, através do qual fluem pacotes de informações de composição conhecida." [DeMARCO, 1978].

Um fluxo de dados deve: [Ashworth e Goodland, 1990] e [Barker, 1990]

- Ter um único nome
- Situar-se entre dois processos, ou
- Situar-se entre um processo e um depósito de dados, ou
- Situar-se entre um processo e uma entidade externa
- Ser constituída de atributos de entidades conhecidas ou, elemento de dado cuja fonte ou destino é clara.

Para [DeMarco, 1978], um fluxo de dados deve obedecer também as regras seguintes:

- Os fluxos de dados que entram e que saem do diagrama-pai, devem ser equivalentes aos que entram e saem no diagrama-filho (Balanceamento).
- Dois fluxos de dados não podem ter o mesmo nome.
- Os fluxos de dados que se movimentam para dentro ou fora dos arquivos não precisam de ter nomes.

2.3.3 CARACTERÍSTICAS

Os DFD's existem há bastante tempo. Foram criados independentemente por várias pessoas, em diversas épocas. O primeiro uso registado foi na França, na década 20. Nessa altura foi usado para reorganizar um escritório cheio de escriturários, com grande burocracia [Meilir, 1988] e, segundo [DeMarco, 1978], esta técnica é conhecida pelos utentes desde 1940 e as suas caractericas são:

1 - Gráficos

Conforme está vincado em quase todas as definições acima, os diagramas são representados em forma de gráficos.

2 - Particionados

A decomposição em níveis mostra claramente esta característica. Na fig. 2.8, está representado um exemplo de particionamento, usando um diagrama de bolhas.

O diagrama 0 ou de contexto, é uma versão não particionada. A partição, no exemplo acima indicado, inicia no diagrama 2, porque o diagrama 1 é "primitivo", ou seja, não resiste a nenhum particionamento.

3 - Multidimensionais

Dão uma visão da imagem total, enquanto que as pessoas, empresas que trabalham com dados, etc, dão uma visão de somente uma parte daquilo que acontece.

4 - Enfatizam fluxos de dados

Os fluxos de dados que são representados no diagrama devem ter informações que sobre eles fluem.

5 - Não enfatizam os fluxos de controle

No diagrama de fluxo de dados, não devem figurar fluxos de dados que documentam a percepção da pessoa que processa os dados, mas sim os fluxos de dados em sí.

Diagrama 1

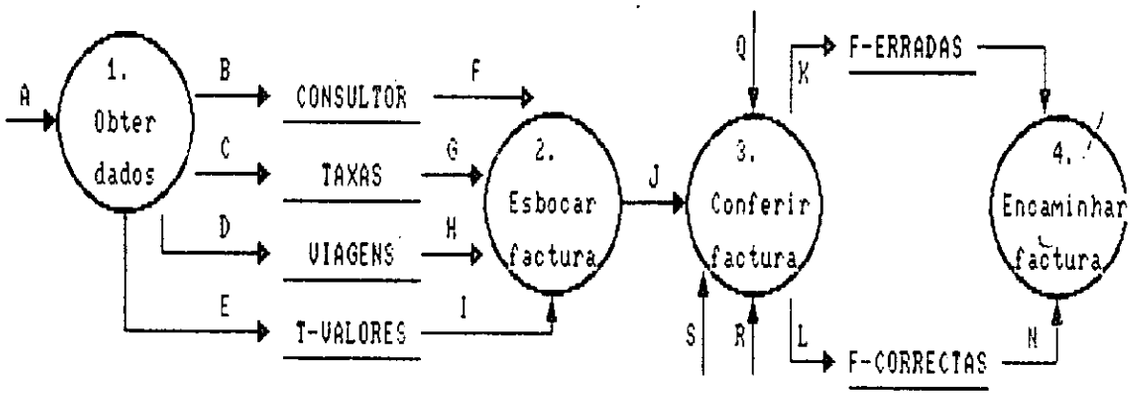


Diagrama 2

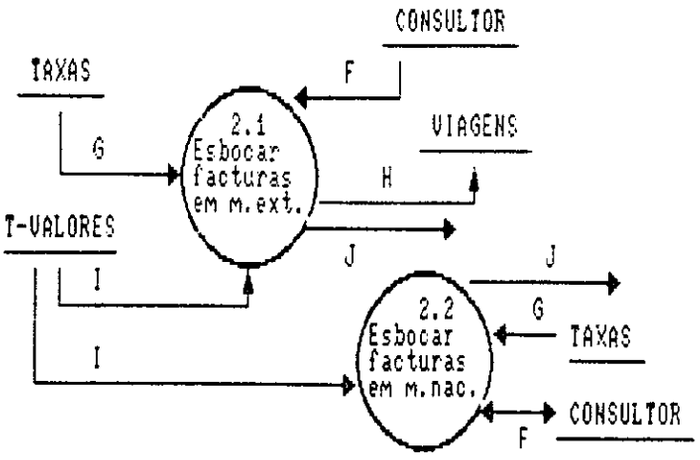


Diagrama 3

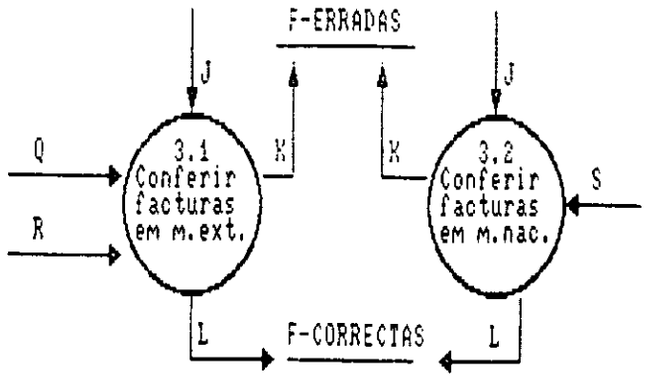


FIG. 2.8 - EXEMPLO DE PARTICIONAMENTO USANDO DIAGRAMA DE BOLHAS

2.3.4 - EXEMPLOS ILUSTRATIVOS

A título de exemplo, ilustra-se a seguir a construção do diagrama de contexto, e em seguida as regra de expansão.

Na metodologia SSADM, quatro conjuntos de DFD's são desenvolvidos:

1 - Físico actual

Nesta fase modela-se o sistema corrente na sua presente implementação.

2 - Estrutura lógica

Do físico actual, extrai-se a representação puramente lógica do sistema actual, isto é, a realização dos processos sem ter em conta quem os executa.

3 - Desenho do sistema

Desenvolvem-se propostas de desenho, cada um satisfazendo os requisitos do novo sistema. Cada desenho é expresso como uma nova visão do DFD, conhecida como opção do sistema.

4 - Requisitos

Usando uma opção de sistema seleccionada e o DFD lógico, desenvolve-se um conjunto completo de DFD, representando o novo sistema.

A relação dos quatro conjuntos é representada na fig 2.9.

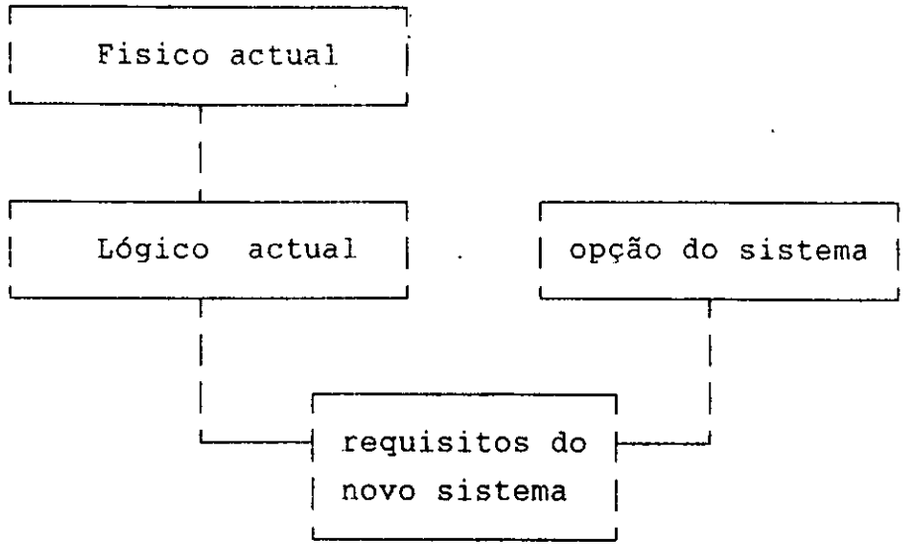


FIG.2.9 - QUATRO CONJUNTOS DE DFD's DESENVOLVIDOS NA METODOLOGIA SSADM

O primeiro exemplo, usando a metodologia SSADM mostra-se na figura 2.10.

No caso da fig. 2.11 a notação usada é a de [Gane e Sarson, 1979] e [Davis, 1987].

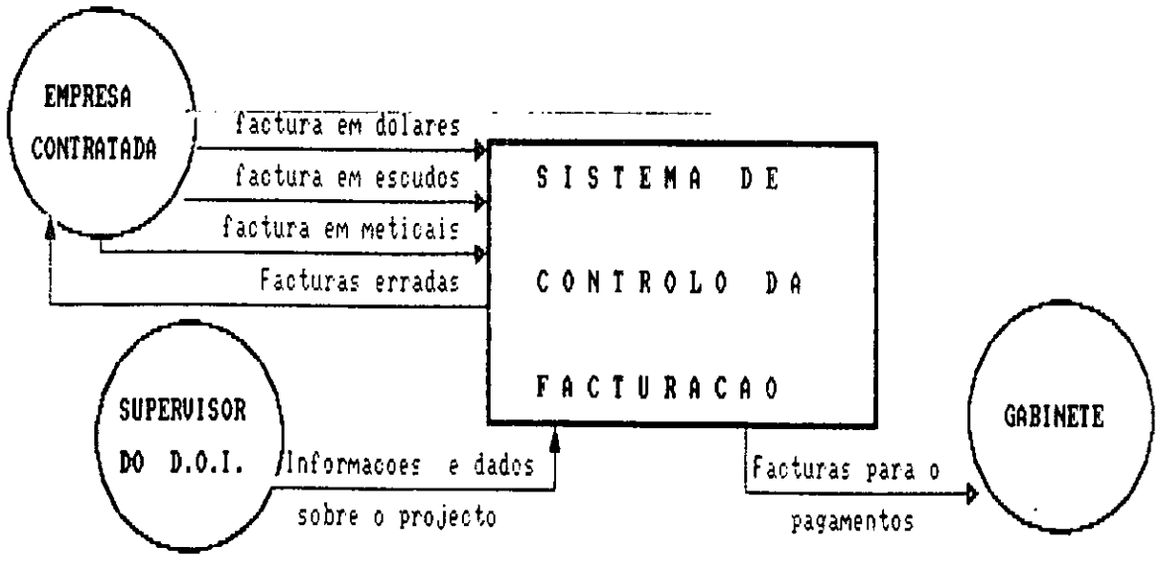


FIG 2.10 - DIAGRAMA DE CONTEXTO, NOTAÇÃO SSADM [Ashworth e Goodland, 1990]

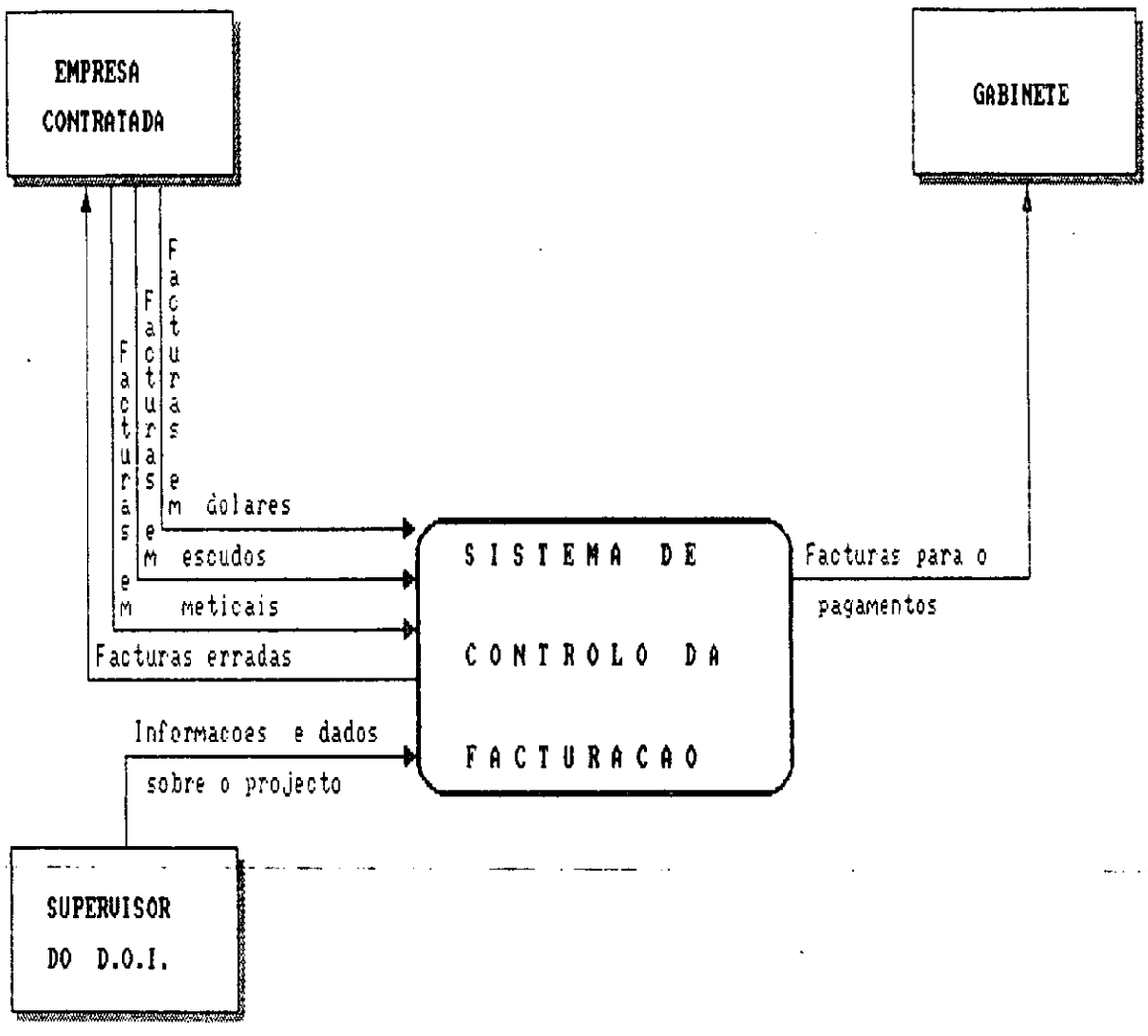


FIG 2.11 - DIAGRAMA DE CONTEXTO, NOTAÇÃO [Gane e Sarson, 1979] e [Davis, 1987]

2.4 - CONVENÇÃO DE EXPANSÃO

Os DFD's do nível inferior resultam da expansão dos processos do nível superior. O seu relacionamento é possível com o uso do número de identificação.

No diagrama de contexto apresentado na fig 2.10 e 2.11, por exemplo, existe apenas um processo, que representa o sistema, ainda a nível de caixa preta e as entidades que com ele se inter-relacionam, (entidades externas).
No nível 1, figuras 2.12 e 2.13, expande-se a caixa preta, dando como resultado os processos 1,2,3 e 4, usando as notações de SSADM [Ashworth e Goodland, 1990] e [Neto et al, 1988], respectivamente.

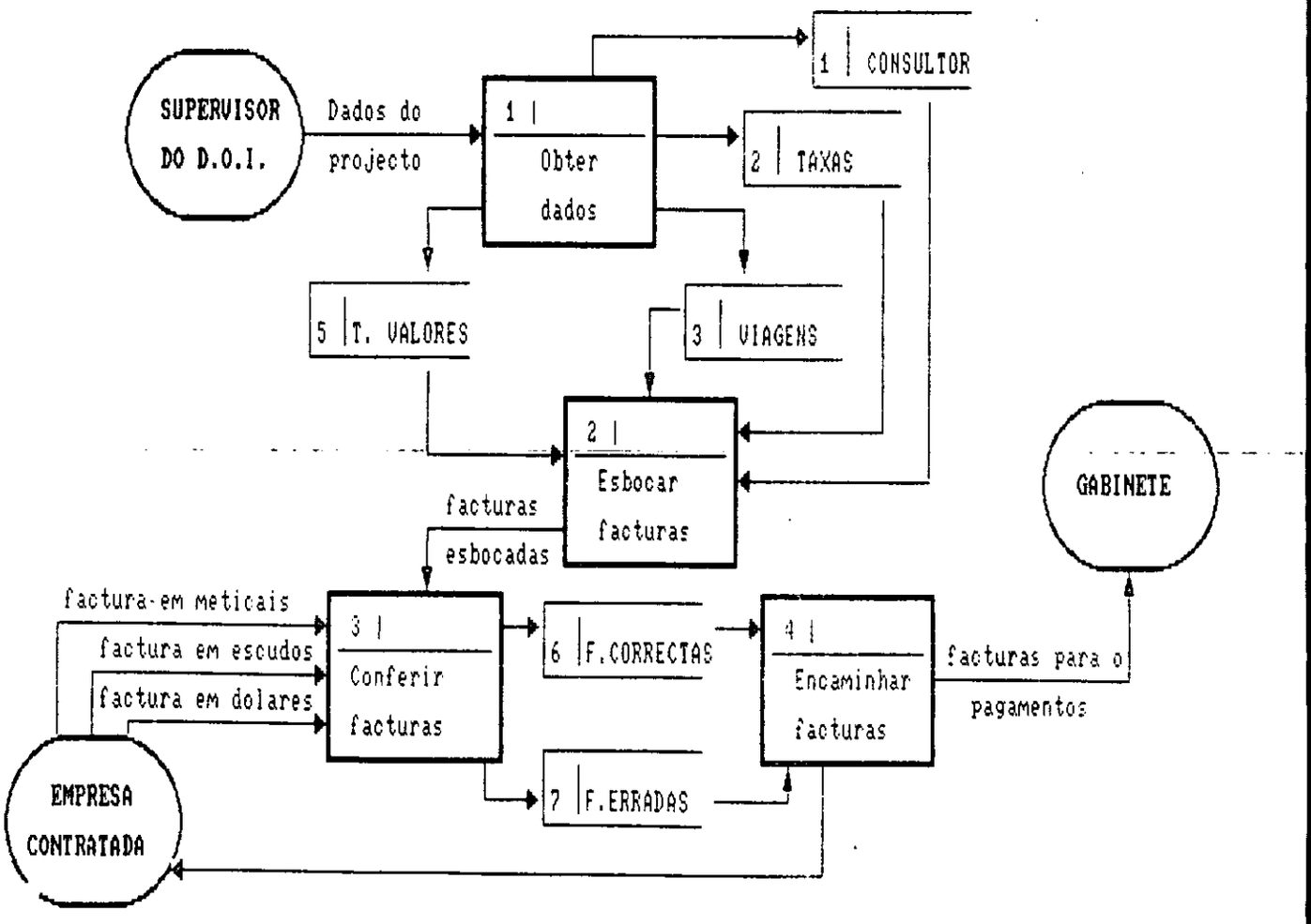


FIG. 2.12 - DIAGRAMA DE NÍVEL 1, NOTAÇÃO SSADM [Ashworth e Goodland, 1990]

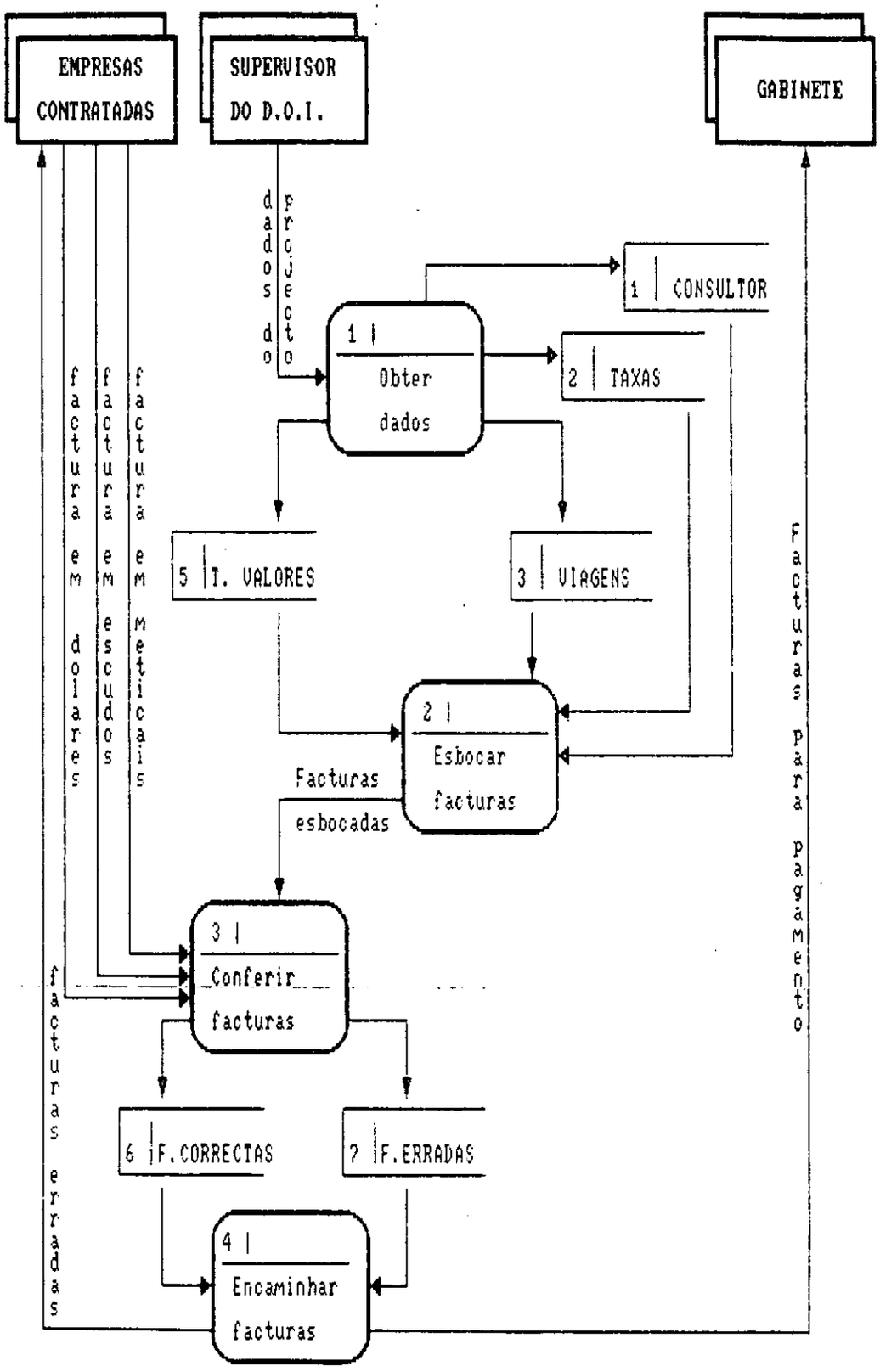


FIG. 2.13 - DIAGRAMA DE NÍVEL 1, NOTAÇÃO [Neto et al, 1988]

A fig. 2.14, mostra a descida de nível de detalhe com a expansão do processo 2, mostrando assim o primeiro DFD do nível 2, uma vez que o primeiro processo não será decomposto em sucessivos DFD's de nível mais baixo, ou seja, o processo 1 é "primitivo funcional", [DeMarco, 1978].

Na fig. 2.15, temos o mesmo diagrama usando a notação desenvolvida por [Meilir, 1988].

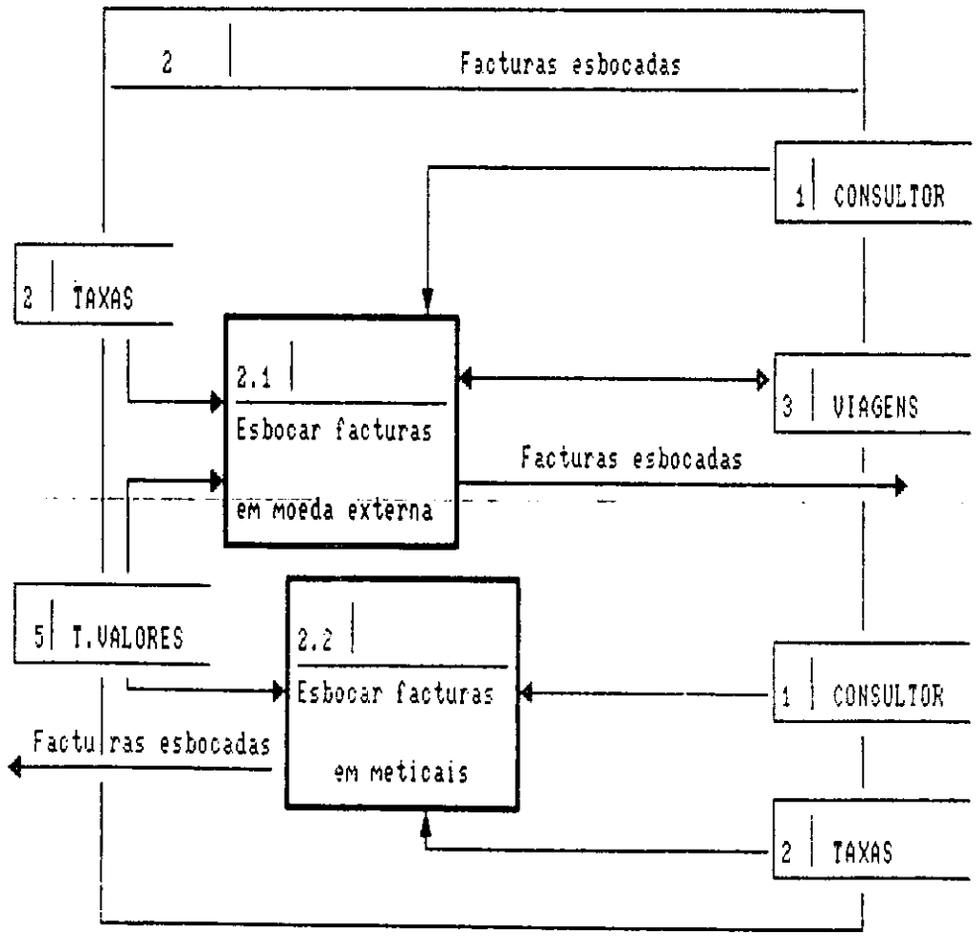


FIG. 2.14 - DIAGRAMA DE NÍVEL 2, NOTAÇÃO SSADM [Ashworth e Goodland, 1990]

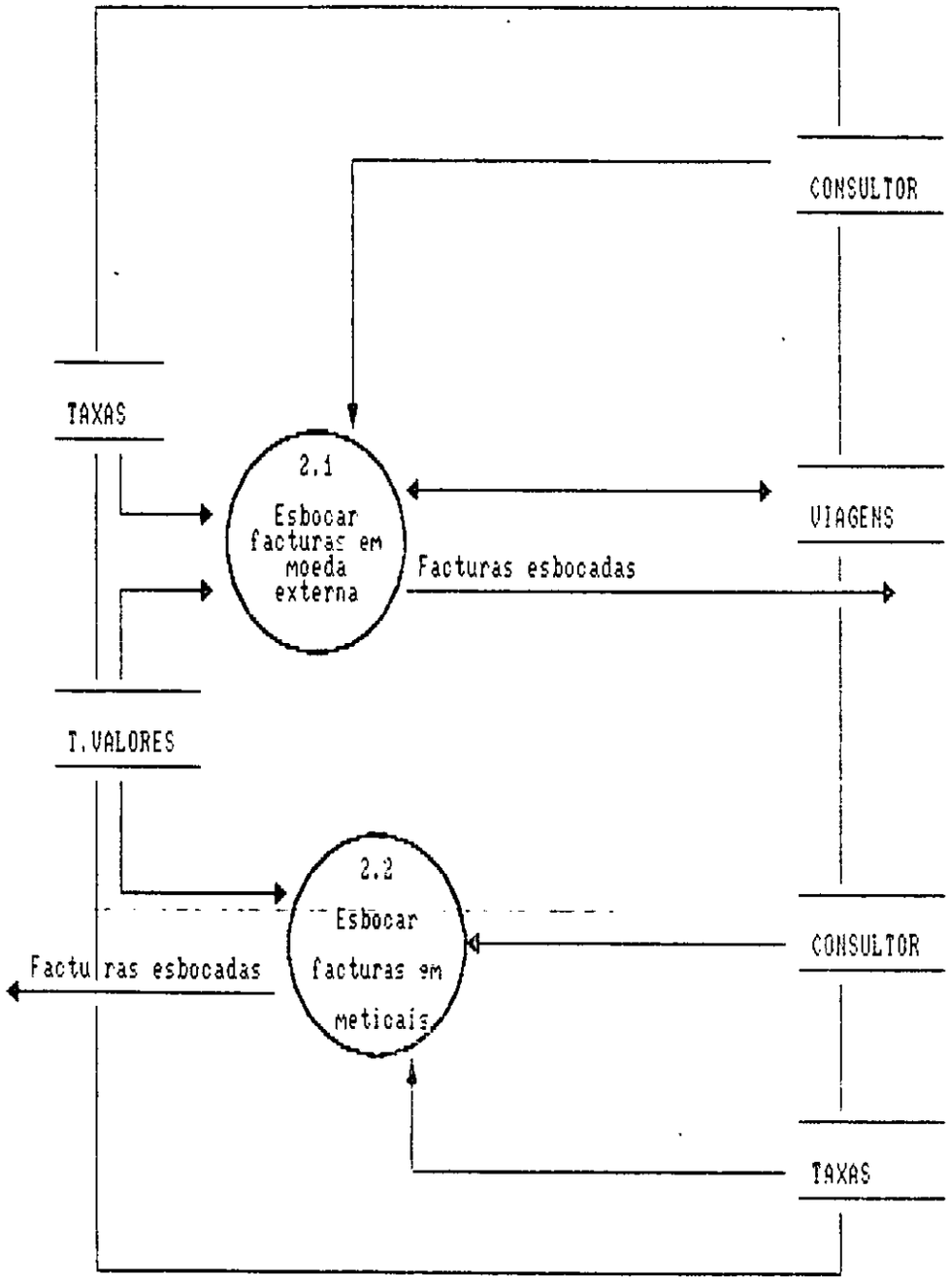


FIG. 2.15 - DIAGRAMA DE NÍVEL 2, NOTAÇÃO [Meilir, 1988]

Do mesmo modo que se expandiu o processo 2, resultando num diagrama de nível 2 (figuras 2.13 e 2.14), pode-se também desdobrar o processo 2.1, 1º DFD do nível 3, obtendo-se os processos 2.1.1 e 2.1.2.

Sendo assim, conclui-se que no 1º nível o número de identificação tem apenas 1 dígito, no 2º nível tem 2 dígitos, no 3º nível 3 dígitos e assim sucessivamente.

Esses dígitos, a partir do nível 2, estão separados por ponto. Estes dígitos são significativos e a sua leitura é da esquerda para a direita.

Tomemos como exemplo o número de identificação 2.1.1. Este número, indica que se trata do 1º processo do DFD resultante da expansão do processo 2.1, do nível 2,(uma vez que, o processo expandido, apresenta dois dígitos, separados por ponto). O processo 2.1 por sua vez, é o 1º processo do DFD resultante da expansão do processo 2 do nível 1.

Esta técnica, como se pode ver, tem um conjunto de regras tendentes a facilitar o mecanismo de análise aos diversos níveis, sem se perder de vista a estrutura do sistema. Contudo, existem também aspectos que constituem desvantagem desta técnica, como se pode ver no ponto seguinte.

2.5 - VANTAGENS E DESVANTAGENS

As vantagens do uso da técnica dos DFD's são:

- É um meio útil e intuitivo de descrever um sistema de forma muito compreensível, tornando possível a sua discussão entre o analista e o usuário, de forma a identificar as falhas atempadamente. É fácil de desenhar ou rectificar, sendo por isso de fácil compreensão, [Ashworth e Goodland, 1990].

- Permite resumir as funções do sistema, [Davis, 1987].

- Permite a geração de soluções alternativas razoáveis, [Davis, 1987].

- Evita ambiguidades, contrariamente ao que acontece nas técnicas informais, baseadas em métodos tradicionais, [Davis, 1987].

- Permite evitar embaraços aos usuários não técnicos, que geralmente ocorrem quando se utiliza a técnica de fluxogramas, devido a simbologia técnica que esta usa, [Davis, 1987].

- Cada nível é útil para o analista porque o detalhe de cada área individual, pode ser investigado isoladamente do resto do sistema, [Ashworth e Goodland, 1990].

- Dá a possibilidade de descrever um sistema a qualquer nível de detalhe, ou seja, cada processo pode ser detalhado separadamente, [Ashworth e Goodland, 1990].

- Permite uma abordagem "top-dowm" para a análise, [De Marco, 1978].

- Não existem conectores reais de continuação em outra

página. Assim cada página é uma apresentação de uma certa àrea de trabalho [De Marco, 1978].

- Da alínea anterior, resulta a possibilidade de, utilizando fotocopiadoras convencionais, poder copiar toda a Especificação Estruturada, [De Marco, 1978].

As desvantagens desta técnica são:

- É sempre necessário descer todos os níveis de detalhe para compreender o funcionamento do sistema.

- Para a obtencão do sistema completo, é necessário um consumo excessivo de folhas, na representação dos processos de cada nível e nos respectivos desdobramentos.

- O uso dos DFD's a partir do físico actual (especialmente por analistas inexperientes) pode causar informações "erradas", pois, a tendência tem sido a de automatizar o físico actual e não de projectar um novo sistema que resolva os problemas do sistema actual.

- A alteração do diagrama no nível mais baixo, origina rectificações sucessivas dos outros diagramas até ao nível mais alto, havendo, por isso, necessidade de usar um CASE TOOL para controlar a consistência.

2.6 CONCLUSÕES

Do estudo aquí feito, resta concluir que o sucesso da técnica dos DFD's reside na participação activa de todos os elementos que fazem parte de uma organização, cujos objectivos pretendem ser alcançados. Conforme vimos, esta é a etapa chave da análise, pois o problema deve ser cuidadosamente estruturado e bem analisado pelos analistas e usuários, de modo que o resultado dos programas ou procedimentos manuais que forem adoptados, sejam, de facto, soluções para os problemas existentes no sistema.

Como forma de evitar que a desvantagem indicada no ponto 2.5 alínea 3 se perpetue e de chamar maior atenção, não só, aos analistas sobre os problemas do sistema físico e o que devem ser os procedimentos no novo sistema depois da depuração, através dos sistemas lógicos actual e proposto, mas também situar melhor os usuários sobre as etapas pelas quais se passa até ao sistema proposto, achamos que devia haver notação específica para cada etapa. Uma proposta, por exemplo, seria usar para o físico actual a notaçao proposta no ponto 2.2, para o sistema lógico actual podia se usar a mesma notação mas colocando no canto superior esquerdo um asterístico, no sistema lógico proposto dois asterísticos e, finalmente, no novo sistema, no lugar dos dois asterísticos, colocar-se-iam as letras "ns" em maiúsculas, significando novo sistema.

A decomposição por níveis de detalhe é também muito importante, porque permite ampliar o ponto de vista sobre o trabalho a ser desenvolvido e, nesse aspecto, esta técnica é de facto uma base sólida para a análise de qualquer sistema.

3 . HIERARCHY / INPUT / PROCESS / OUTPUT (HIPO)

3.1 - DEFINIÇÕES

HIPO é um conjunto de diagramas que descrevem gráficamente funções do geral para o nível de detalhe. Baseia-se na representação gráfica das funções, sua organização aninhada de detalhes crescentes, e representa items de dados de entrada e saída em cada nível [IBM, 1975].

Para [Davis, 1987] HIPO é um pacote composto por duas partes: os quadros hierárquico e IPO.

O quadro hierárquico representa a estrutura do programa de cima para baixo ou seja hierárquicamente. O quadro IPO (Input/Process/Output) é usado para descrever entradas, processos de cada módulo representado no quadro Hierárquico e saídas.

3.2 - FORMAS DE APLICAÇÃO

Dada a sua natureza gráfica, pode ser usado:

- desde o início da análise através da implementação, da escrita e da manutenção ou modificação de programas, pela fácil identificação do código, [IBM, 1975];

- pelos usuários e administradores para seguir com facilidade a estrutura do programa, enquanto que os programadores usam-no na escrita, manutenção ou modificação do programa, [Davis, 1987].

O diagrama HIPO [IBM, 1975] tem o formato mostrado na fig. 3.1 e na fig. 3.2, mostra-se o formato do quadro IPO segundo [Davis, 1987].

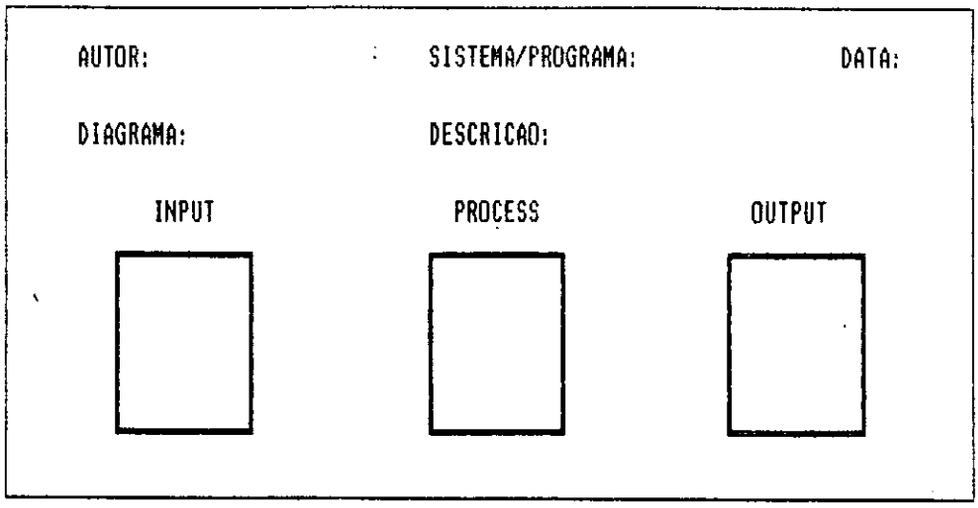


FIG. 3.1 - DIAGRAMA HIPO

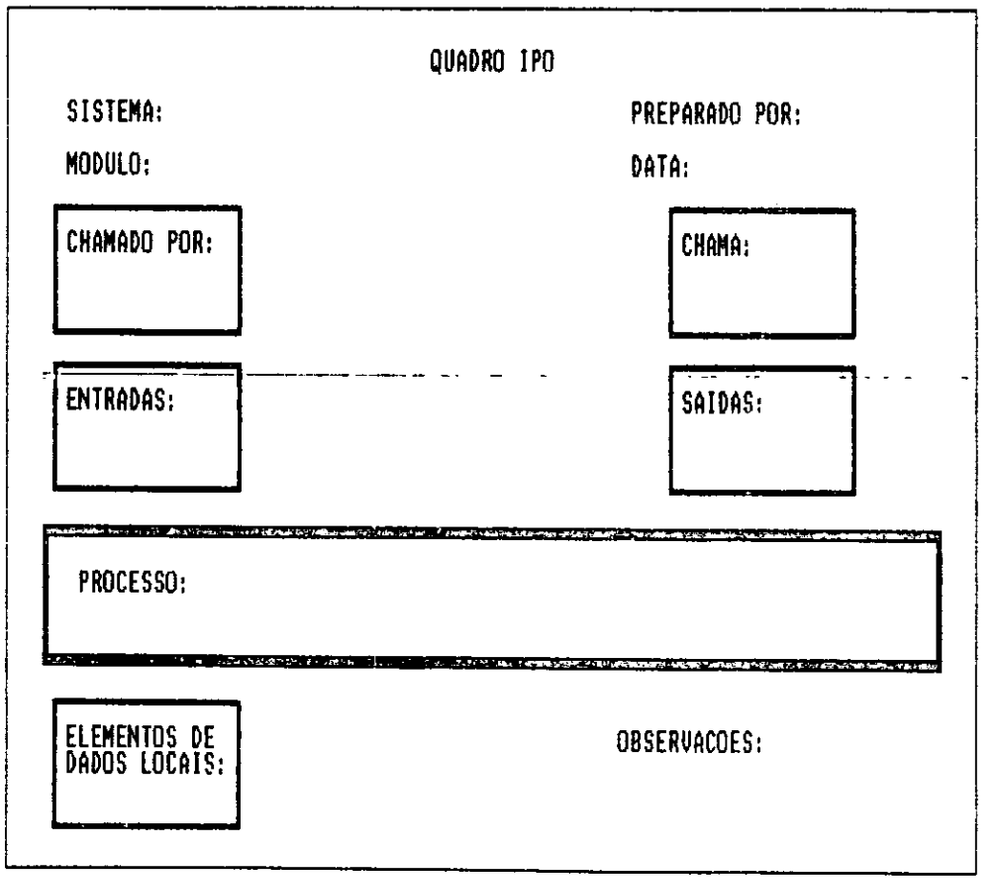


FIG. 3.2 - QUADRO IPO

HIPO permite que as transformações dos dados de entrada em dados de saída sejam visíveis. Assim, a secção "input" - contém items de dados usados pela fase do processo, a secção "process" contém uma série de etapas enumeradas que descrevem a função feita e a secção "output" providencia a saída produzida por cada função para cada nível do diagrama.

3.3 - DESCRIÇÃO

Segundo [IBM, 1975], existem 3 tipos de diagramas no pacote típico HIPO:

- Tabela Visual de Conteúdos ou Quadro Hierárquico
- Diagrama Global ou Diagrama de nível mais alto
- Diagramas Detalhados ou Quadros IPO

1 - Tabela Visual de Conteúdos

Contém nomes e números de identificação de toda a visão e detalhe do diagrama, mostra a sua estrutura e relacionamento das funções na forma hierárquica. Da-se, como exemplo a fig. 3.3.

O quadro hierárquico concebido por [Davis, 1987], não contém a numeração dos módulos, sómente a disposição dos módulos representa gráficamente a hierarquia do programa, fig 3.4.

O quadro hierárquico é desenvolvido de cima para baixo, e por conseguinte, os módulos de alto nível são genéricos e realizam funções de controle enquanto que os de baixo nível ocupam-se dos cálculos detalhados. Para a obtenção dos dados detalhados, os módulos de alto nível são decompostos conforme sejam coesivos ou funcionais. O módulo diz-se "Coesivo" se realiza uma única função lógica completa. Chama-se "Decomposição Funcional" a decomposição dum módulo em suas subtarefas.

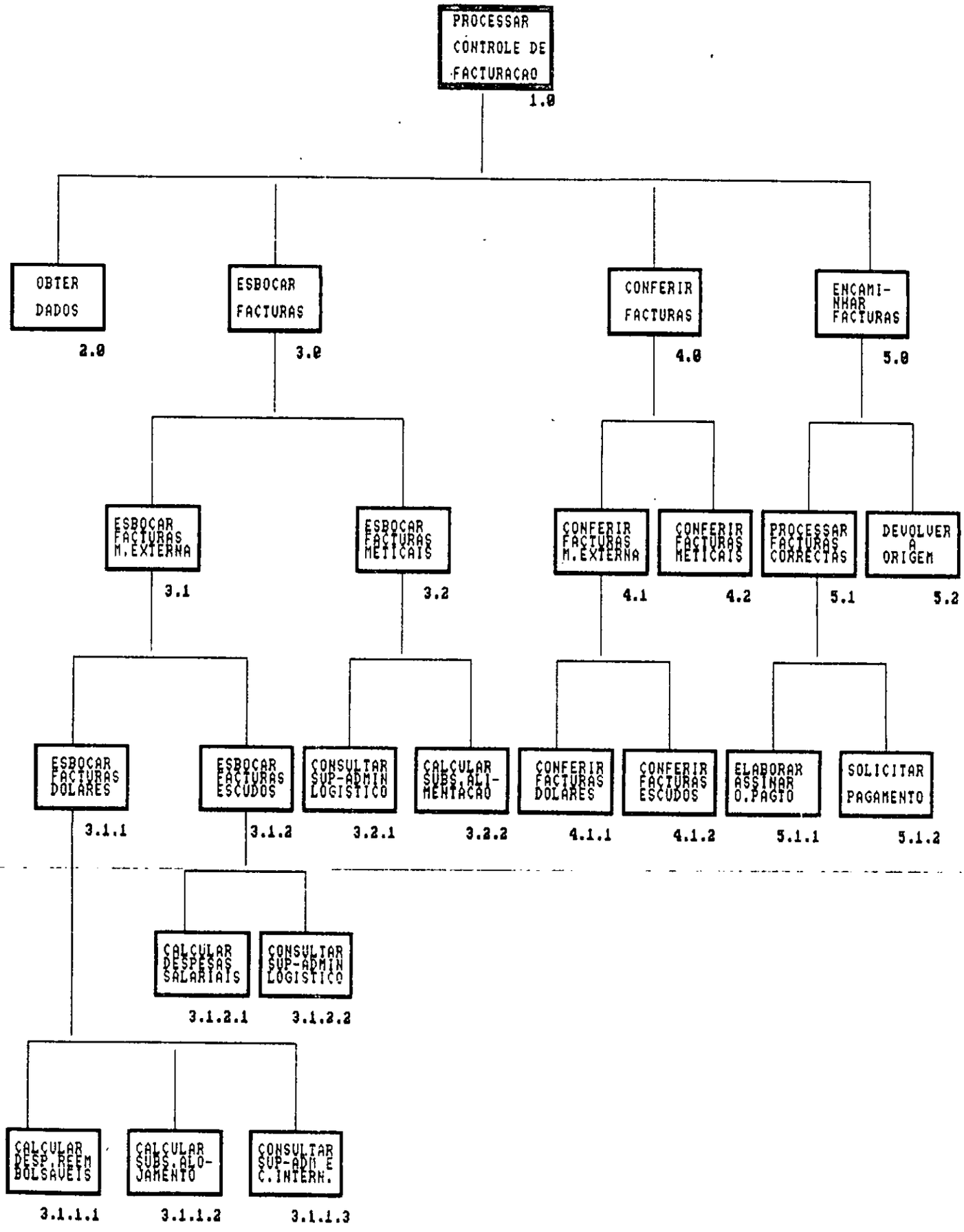


FIG. 3.3 - TABELA VISUAL DE CONTEÚDOS

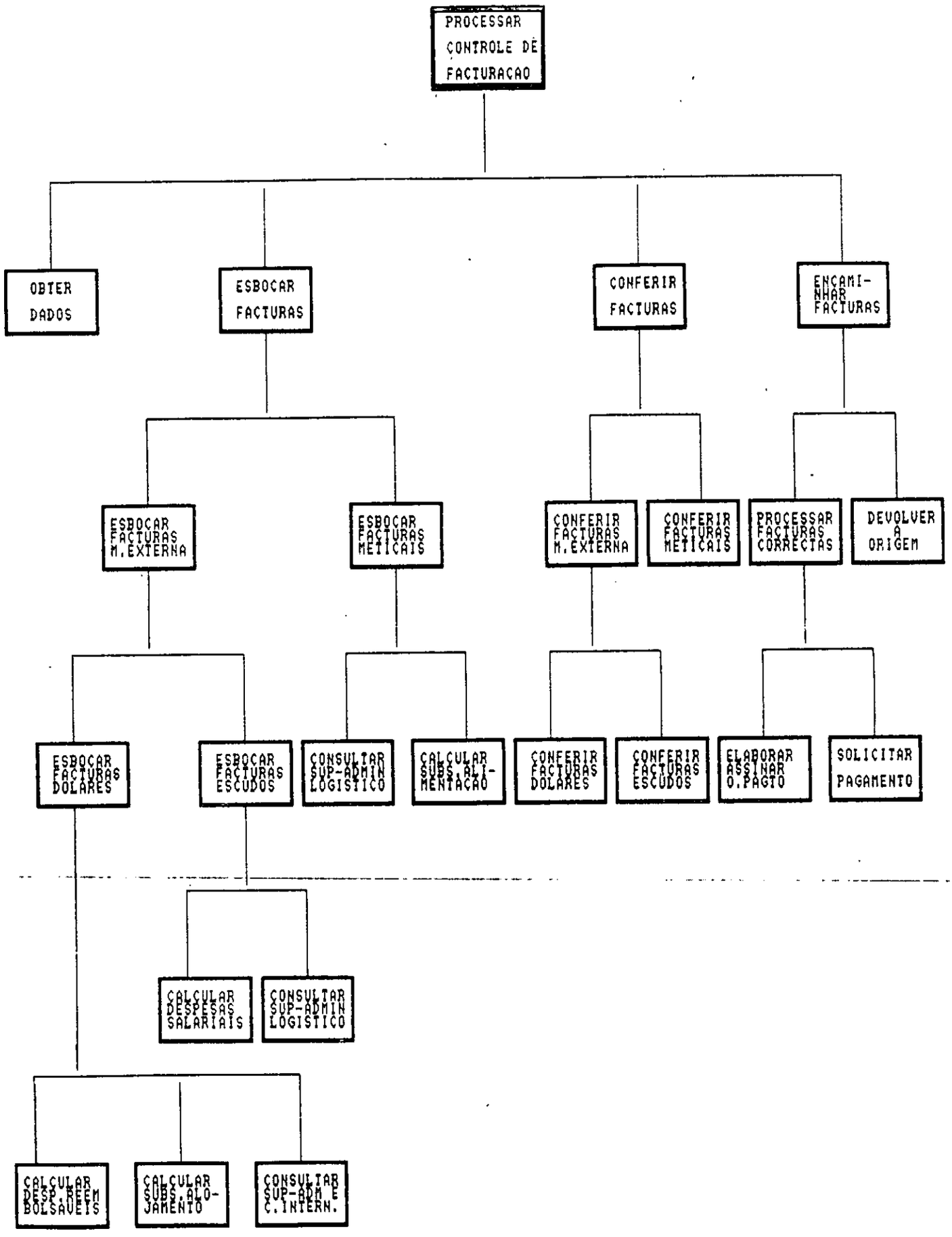


FIG. 3.4 - QUADRO HIERÁRQUICO

2 - Diagrama Global

Este descreve as funções principais e referencia as necessidades do diagrama detalhado para expandindo as funções para um detalhe suficiente. Proporciona em termos gerais entradas, processos e saídas. Os items de dados de entrada e os de saída são conectados com as respectivas fases do processo através de setas, fig 3.5.

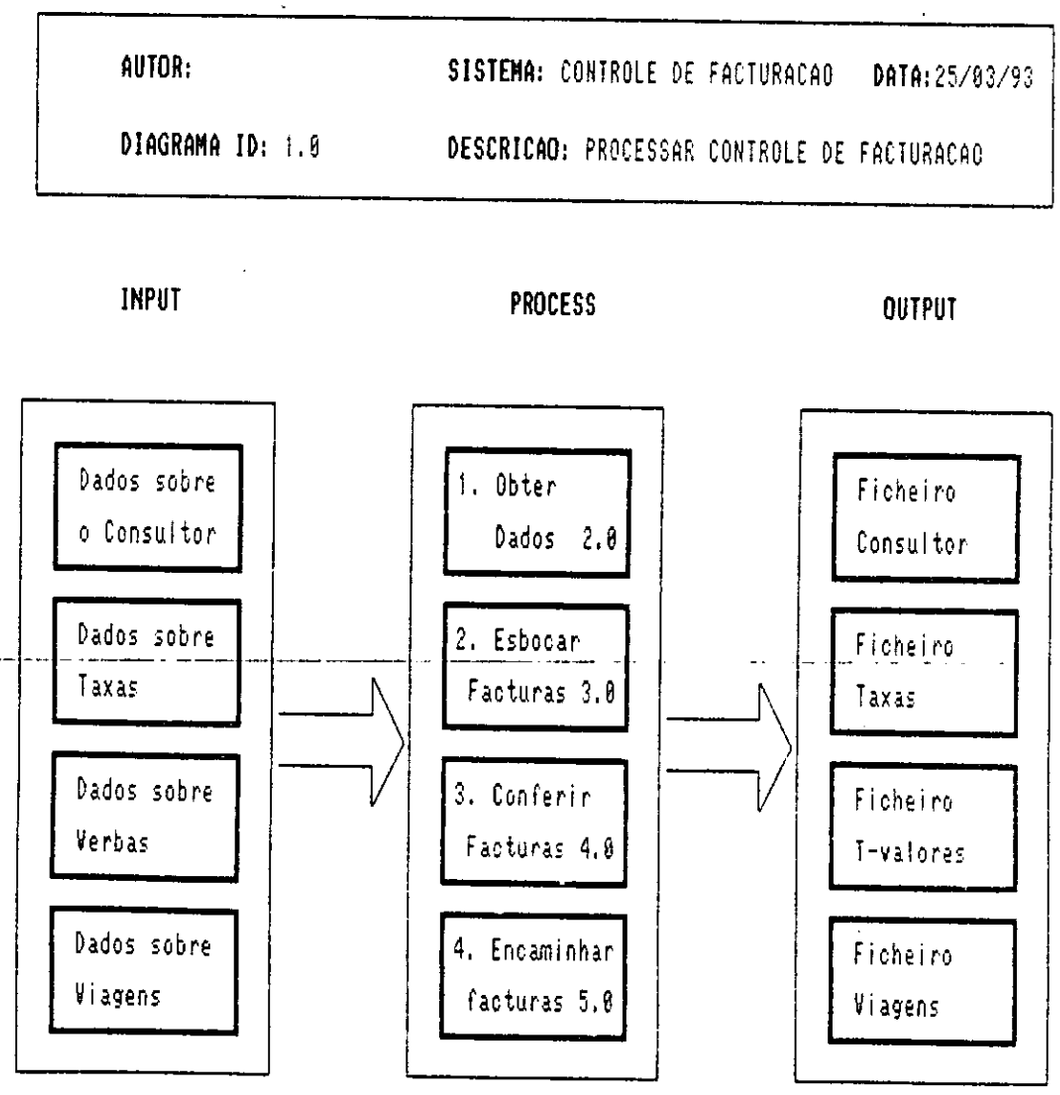


FIG. 3.5 - DIAGRAMA GLOBAL

3 - Diagrama Detalhado

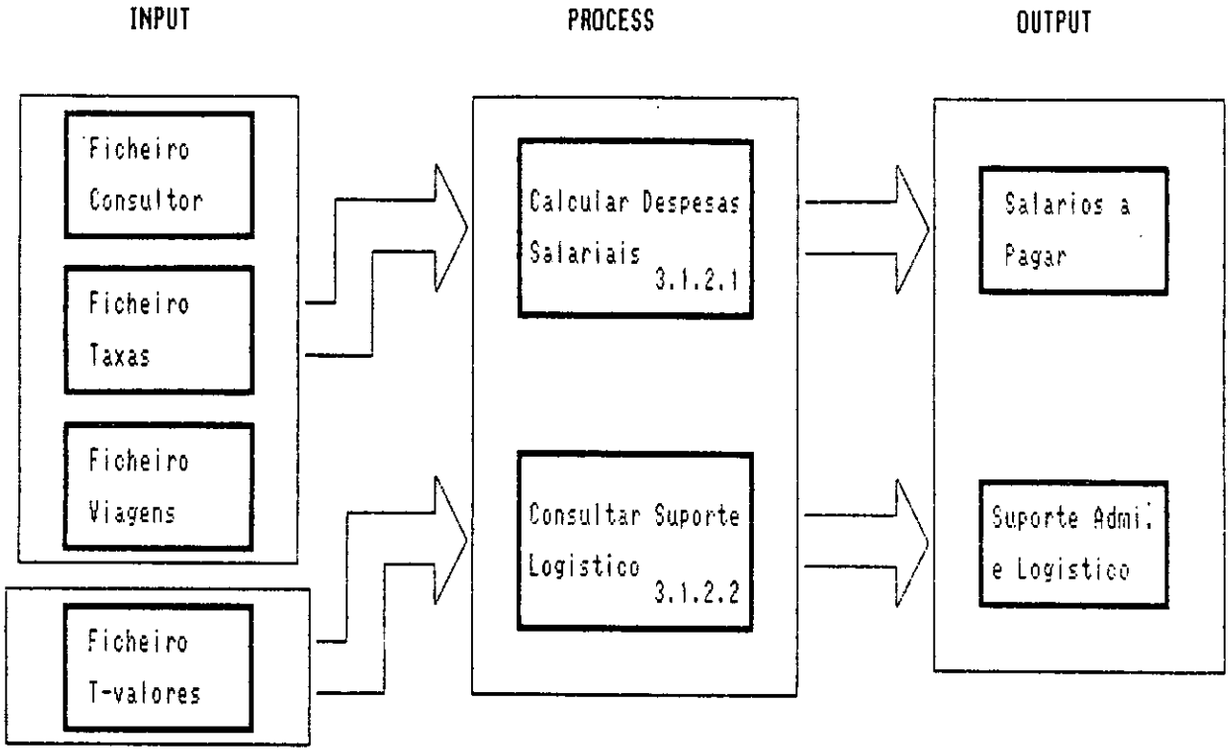
Descreve as funções específicas, mostra items específicos de entrada e saída e referencia outros diagramas detalhados. Este contém para além das secções de entrada, processo e saída, a secção de "descrição detalhada" a qual amplia as etapas do processo e referencia o código associado com a fase do processo. Esta secção referencia também diagramas HIPOs de nível baixo. O número de níveis de detalhe do diagrama depende da complexidade do diagrama e da quantidade de informação a documentar, fig. 3.6.

O quadro IPO, tem na sua parte superior a identificação do sistema, módulo, autor e data. A parte central está reservada a identificação dos módulos chamador e chamado, os dados de entrada e saída bem como a descrição dos processos(cujos passos são definidos em Português Estruturado). A parte inferior tem uma área para elementos de dados locais[1] e anotações, [Davis, 1987].

A fig. 3.7 mostra o quadro IPO para o módulo apresentado na fig. 3.6.

[1] é um elemento de dados utilizado apenas dentro de um único módulo [Davis, 1987, p.84]

AUTOR:	**SISTEMA:** CONTROLE DE FACTURACAO **DATA:**25/03/93
DIAGRAMA ID: 3.1.2	**DESCRICAO:** ESBOCAR FACTURAS EM ESCUDOS



NOTAS	MODULO	SEGMENTO	REF.
1. Se o consultor efectuou viagens, Achar o tempo para calcular o salario consoante a estadia seja em Portugal ou em Mocambique. Se o consultor nao viajou Calcular o seu salario conforme a taxa "Servico em Portugal" ou "Servico em Mocambique"	ID 3.1.2		3.1.2.1
2. Ler o ficheiro T-valores Se chave = SUP_LOGESC Escrever valor correspondente	ID 3.1.2		3.1.2.2

FIG. 3.6 - DIAGRAMA DETALHADO

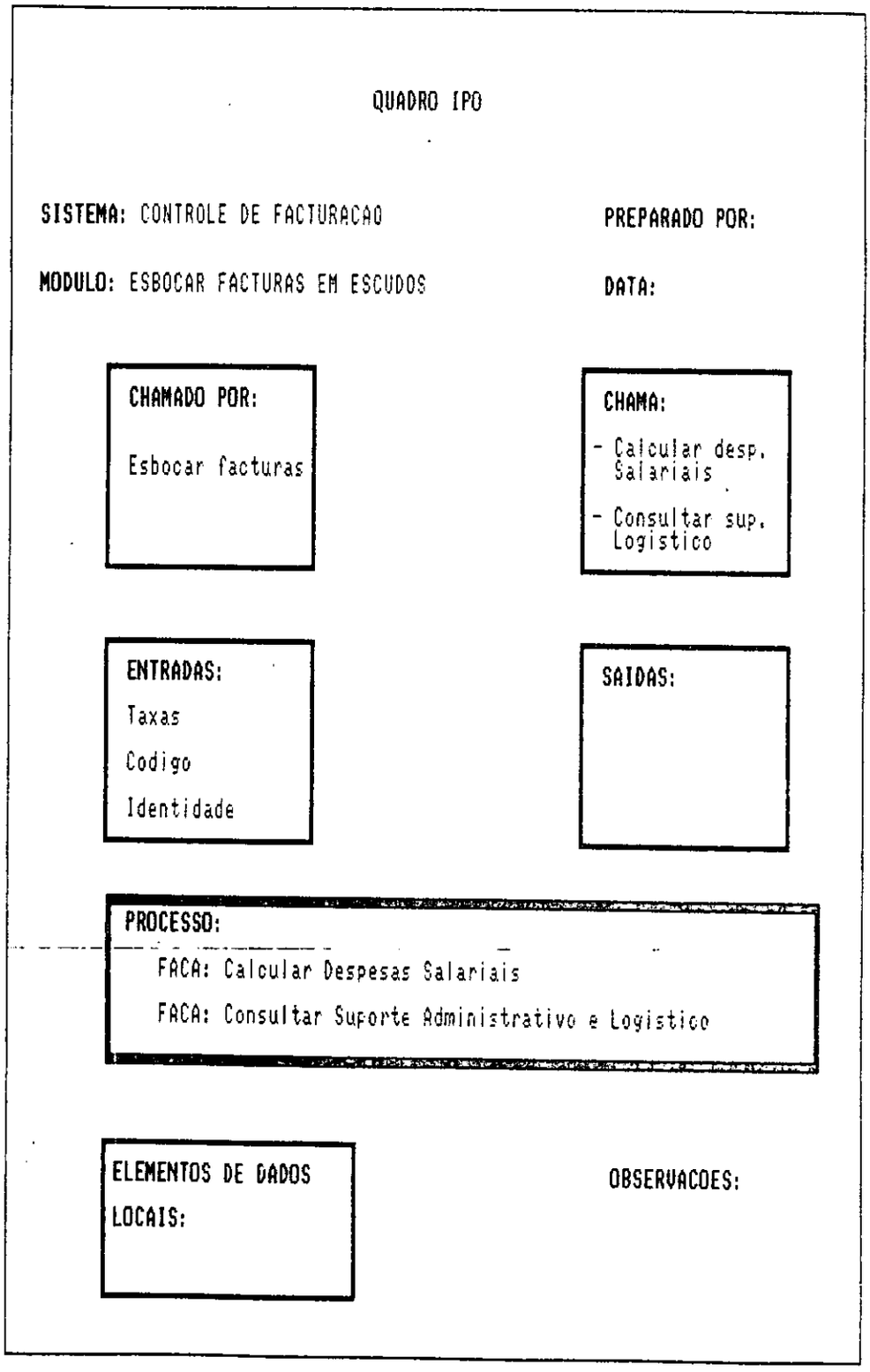


FIG. 3.7 - QUADRO IPO

3.4 - VANTAGENS E DESVANTAGENS

As vantagens da técnica HIPO são:

- Proporcionar uma estrutura pela qual as funções do sistema podem ser entendidas, [IBM, 1975];
- Dividir sistemas complexos ou programas em partes manejáveis, [IBM, 1975];
- Apresentar as funções a realizar pelo programa antes que se especifiquem as declarações do programa, [IBM, 1975];
- Avaliar, refinar o projecto e corrigir falhas antes da implementação, [Davis, 1987];
- Facilitar a manunteção, [Davis, 1987];
- Possibilitar que os quadros IPO sejam embutidos dentro do código-fonte como uma série de comentários, [Davis, 1987].

As desvantagens da técnica HIPO são:

- A modificação do programa modifica também a documentação;
- A forma de validar e processar dados é mostrada apenas quando o quadro IPO fôr convertido em pseudo-código, [Davis, 1987].

3.5 - CONCLUSÕES

Esta técnica que se enquadra na fase do desenho detalhado, permite representar o desenho de um programa completo.
A documentação HIPO, facilita não só a subdivisão de sistemas ou programas complexos em partes manejáveis mas também a percepção pelos usuários, administradores e programadores das funções a realizar.

A maior clareza sobre o funcionamento de um sistema é possível nesta técnica com o uso do quadro hierárquico, no qual os módulos a realizar são ilustrados em caixas pretas e cujo detalhe é feito em quadros IPO, que são elaborados para cada módulo. A possibilidade de embutir o quadro IPO dentro do código-fonte como uma série de comentários é, talvez a forma mais fácil de documentação de programação estruturada, [Davis, 1987].

4 - FLUXOGRAMA

4.1 - DEFINIÇÕES

Segundo [Davis, 1987], pode-se referenciar dois tipos de fluxogramas: - Fluxograma de lógica do programa e fluxograma do sistema. O primeiro é uma representação gráfica da lógica do programa, enquanto que o segundo, é uma ferramenta tradicional para descrever um quadro de alto nível do sistema físico.

Para [Meilir, 1988], um fluxograma lógico é uma visão do mundo através de um processador imaginário. É uma lista de "primeiro faça isto, depois isto ... e finalmente isto", exactamente um passo de cada vez, numa ordem bem definida.

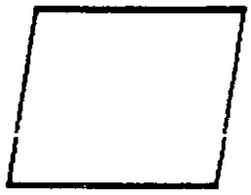
4.2 - SIMBOLISMO

Embora [Davis, 1987] referencie dois tipos de fluxogramas, alguns dos seus símbolos são comuns. Para fluxogramas de lógica do programa, [Davis, 1987] apresenta quatro símbolos, enquanto [Marques, Guedes, 1990], inclui também os dois últimos, conforme ilustrado na fig. 4.1.

Ponto terminal - Marca o começo ou término do programa

Processo - Representa qualquer operação

Decisão - Indica uma decisão sim/não a ser feita pelo programa



Entrada/saída - Representa qualquer operação de entrada ou saída

Conector - Une dois extremos de um fluxograma, geralmente separados por uma página

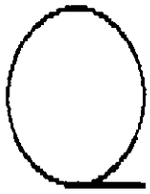
Direcção do fluxo - Representa a direcção do fluxo

FIG. 4.1 - SÍMBOLOS DOS FLUXOGRAMAS DE LÓGICA DO PROGRAMA

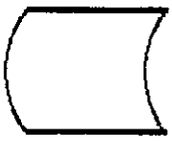
Excluindo o símbolo de decisão, os restantes símbolos são usados no fluxograma de sistema, juntamente com os indicados na fig. 4.2

Conector da outra página
Indica uma saída ou entrada de outra página do fluxo

Cartão perfurado - Entrada ou saída que use cartão perfurado. Também um arquivo de cartão perfurado

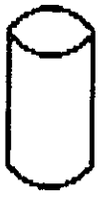


Fita magnética - Entrada ou saída por fita magnética

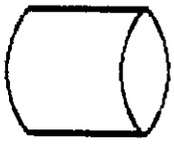


Armazenamento on-line - Símbolo generalizado para qualquer tipo de armazenamento on-line, incluindo disco, tambor magnético, dispositivo de armazenamento de massa, disquete, etc.

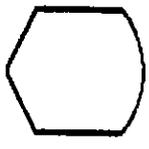
Documento - Normalmente usado para a saída impressa. Também pode designar a entrada de dados através do terminal de impressão.



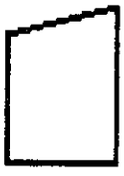
Disco magnético - Entrada ou saída por disco magnético, ou um arquivo ou base de dados armazenados em disco magnético.



Tambor magnético - Entrada ou saída do tambor magnético, ou um arquivo ou base de dados, armazenados em tambor magnético.

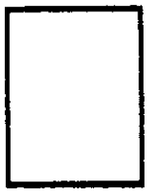


Tela - Terminal ou unidade de apresentação semelhante, pode ser utilizado para identificar a entrada, saída ou ambos.



Entrada manual - Processo off-line, que exija a entrada de dados manual, por exemplo, o preenchimento do formulário ou a marcação em cartão sensível.

Operação manual - processo realizado por meios manuais, p.e., assinatura de cheques.



Operação auxiliar - Operação off-line que utiliza equipamento, p.e., a tranferência de dados da fita de leitura para a fita magnética, para posterior entrada no computador.

Ligação de comunicação - A transição de dados através de uma linha de comunicação.

FIG. 4.2 - SÍMBOLOS ADICIONAIS DOS FLUXOGRAMAS DE SISTEMA

4.3 - FORMAS DE APLICAÇÃO

4.3.1 - Fluxograma de sistemas

O fluxograma de sistemas representa, a nível de caixa preta, cada componente discreto do sistema - programas, arquivos, formulários, procedimentos, etc. Tomemos como exemplo um sistema de pagamento de facturas ilustrado na figura 4.3. Este, mostra o ponto inicial, as entradas e saídas do Sistema de controlo de facturação (representado por uma caixa preta), os procedimentos manuais, os documentos de saída que estes procedimentos produzem, e finalmente o ponto terminal.

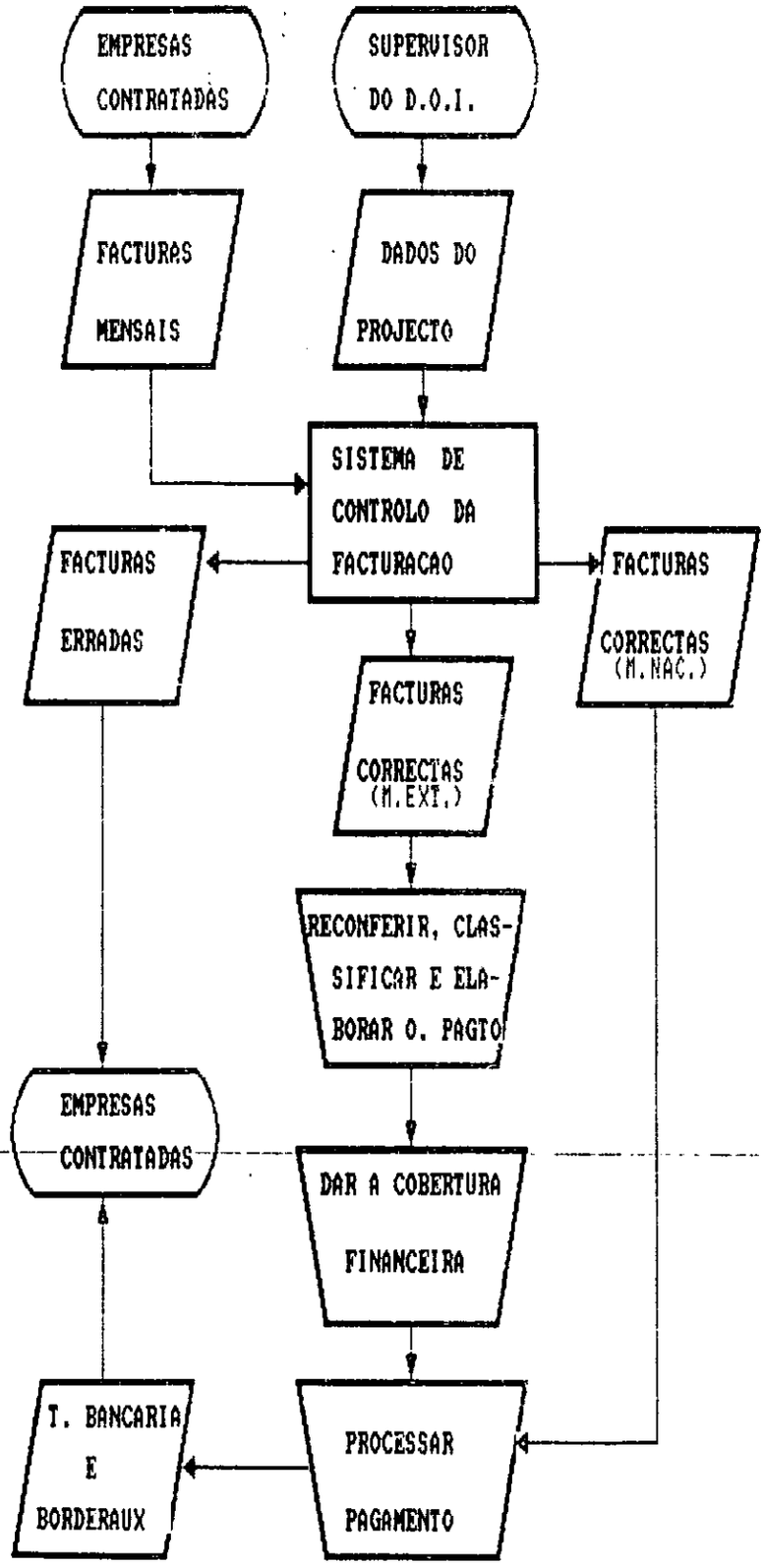


FIG 4.3 - FLUXOGRAMA DO SISTEMA DE PAGAMENTO

4.3.2 - Fluxograma da lógica do programa

Em estruturas "top-down", onde são usados fluxogramas, os programas são desenvolvidos módulo a módulo.

Tomemos como exemplo a figura 2.4 (Diagrama de nível 1). O primeiro processo deste DFD cria quatro (4) depósitos de dados: Taxas, Consultores, T-valores e Viagens. Estes depósitos de dados são criados através de módulos.

A lógica de programa pode ser expressa como combinação de três construções básicas: sequência, decisão e repetição
ilustradas nas figuras 4.4, 4.5-4.7 e 4.8-4.9 , respectivamente.

A construção sequência consiste na realização de um ou mais programas de acção um após outro, sem interrupção.

A construção de decisão, que se pode apresentar de várias formas, está representada graficamente e em forma algorítmica.

A construção de decisão consiste em dois ou mais programas de acção subordinadas, apenas um deles se aplica em qualquer dos casos. [De Marco, 1978, p.68]

A construção de repetição tem duas lógicas possíveis: A lógica "Fazer enquanto" e a lógica "Fazer até que", que correspondem as estruturas "Do While" e "Repeat".

A característica comum das três construções é possuirem único ponto de entrada e único ponto de saída.

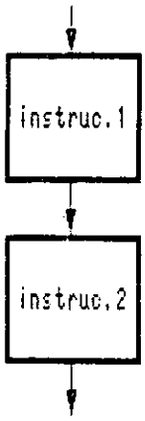


FIG. 4.4 - A CONSTRUÇÃO SEQUÊNCIA

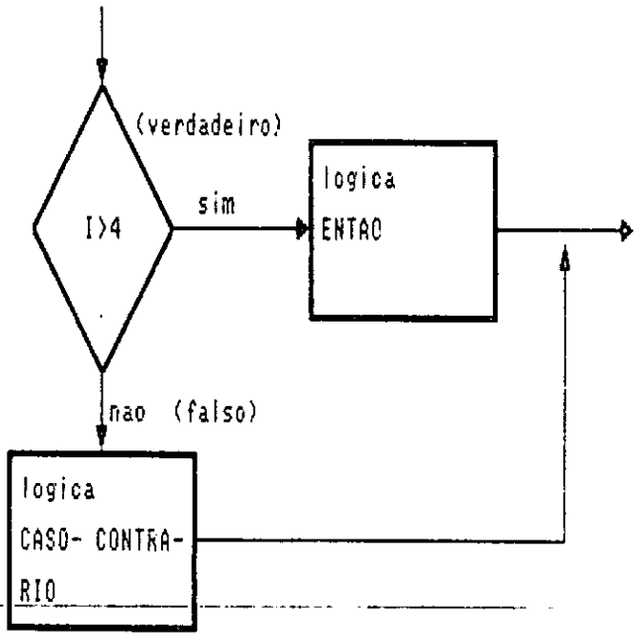


FIG. 4.5 - A CONSTRUÇÃO DECISÃO

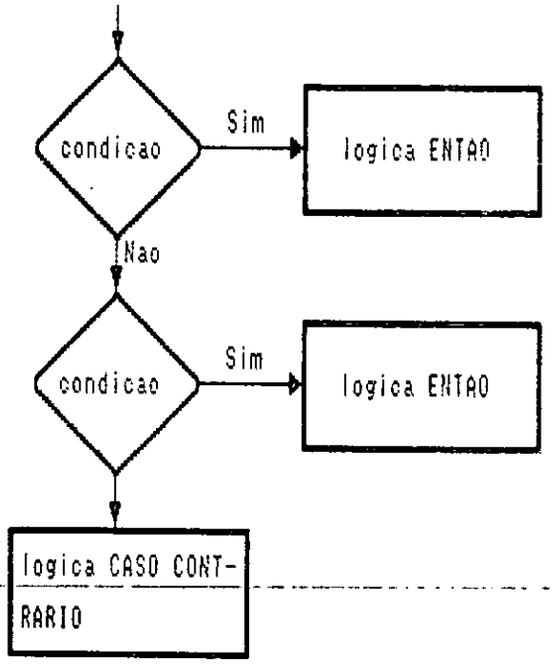
O bloco de decisão pressupõe a lógica SE-ENTÃO-CASO CONTRÁRIO, fig.4.5, cuja estrutura lógica é a seguinte:

```
IF <condição> THEN
     <instrução>
ELSE
     <instrução>
ENDIF
```

O bloco de decisão contém dois blocos de sequência, uma para a lógica ENTÃO e o outro para a lógica CASO CONTRÁRIO.

No fluxograma de alto nível, o bloco pode representar uma sub-rotina, contendo lógicas de sequência, decisão e repetição.

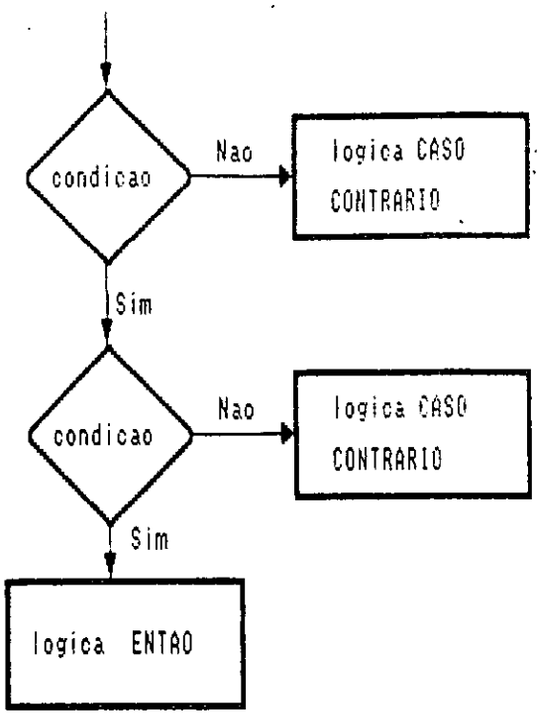
A lógica de decisão pode ser aninhada também no ramo CASO CONTRÁRIO, e o aninhamento pode ocorrer em ambos os caminhos, como se ilustra nas figuras 4.6 e 4.7,seguidas das respectivas estruturas lógicas.




```
IF <condição> THEN
     <instrução>
ELSE
     <instrução>
     IF <condição> THEN
          <instrução>
     ELSE
          <instrução>
     ENDIF
ENDIF
```

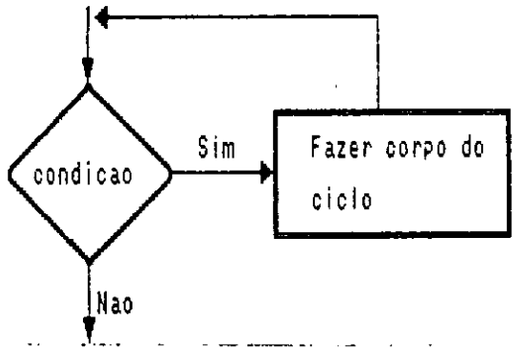
FIG. 4.6 - CONSTRUÇÃO DECISÃO (1a VERSÃO DE ANINHAMENTO)

O caso podia dar-se também de forma inversa, conforme se ilustra na fig. 4.7



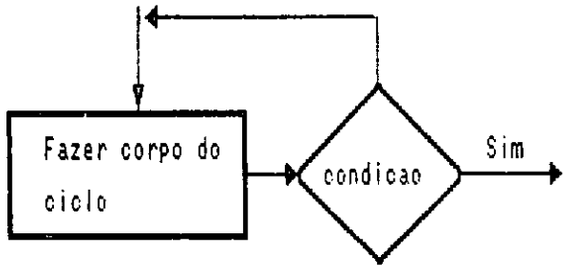

```
IF <condição> THEN
      IF <condição> THEN
            <instrução>
      ELSE
            <instrução>
      ENDIF
ELSE
      <instrução>
ENDIF
```

FIG. 4.7 - CONSTRUÇÃO DECISÃO (2a VERSÃO DE ANINHAMENTO)




```
WHILE <condição>
 DO <instrução>
ENDDO
```

FIG. 4.8 - A CONSTRUÇÃO REPETIÇÃO (FAZER ENQUANTO)




```
REPEAT<instrução>
UNTIL<condição>
```

FIG. 4.9 - A CONSTRUÇÃO REPETIÇÃO (FAZER ATÉ QUE)

Depois desta breve explicação sobre a forma de aplicação, acha-se ser mais elucidativo apresentar um exemplo prático, fig. 4.10, que mostra a lógica do programa de inserção de dados iniciais no ficheiro TAXAS.

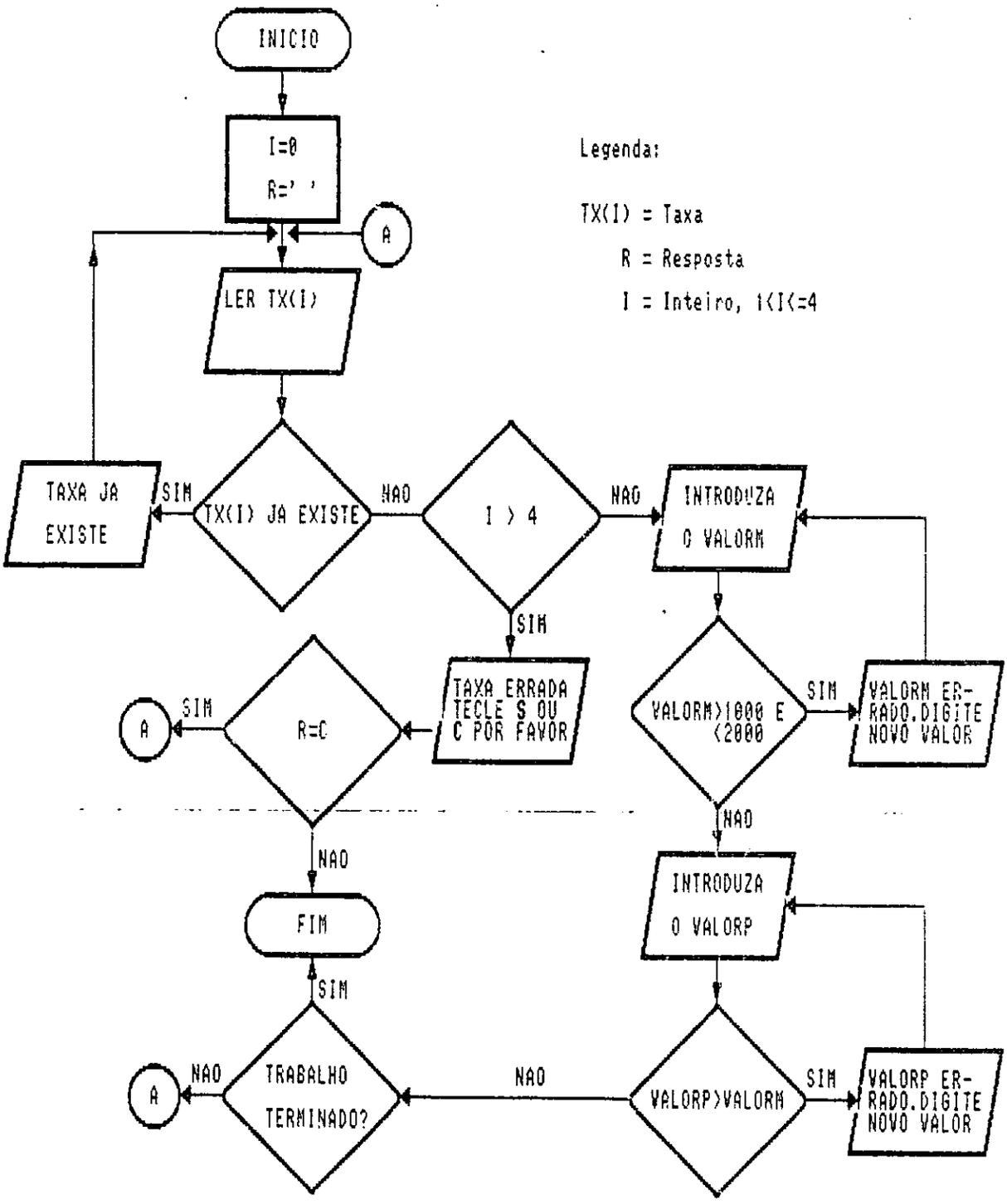


FIG. 4.10 - FLUXOGRMA LÓGICO DO PROGRAMA DE INSERÇÃO DE DADOS NO FICHEIRO TAXAS

4.3.3 - Decomposição de um fluxograma

Um fluxograma pode ser decomposto através do uso de técnicas estruturadas, preservando-se para isso a topologia do fluxograma original. A decomposição do fluxograma em subfluxogramas facilita a estruturação individual dos módulos, além de permitir a divisão do trabalho pelos programadores. Desta decomposição obtém-se um fluxograma na forma canónica, onde processos independentes dos programas são claramente delineados,[Prather e Giulieri, 1981].

[Bohm e Jacopini, 1966], mostram que "para cada fluxograma, existe um fluxograma "estruturado" equivalente".

Tomemos como exemplo a fig.4.10 (fluxograma lógico do programa de inserção de dados no ficheiro Taxas). Elabora-se primeiro o fluxograma esquemático, fig. 4.11 (a) e depois o fluxograma associado fig. 4.11 (b).

Uma melhor compreensão dos procedimentos da decomposição dos fluxogramas é dada por algorítmos que, por sua vez, se baseiam em teoremas e proposições, demonstrados em [Bohm e Jacopini, 1966].

Teorema 1

Todo o fluxograma F tem uma decomposição sequencial única, F= f1of2o...ofm, em subfluxogramas sequencialmente irredutíveis.

Teorema 2

Seja F um fluxograma sequencialmente irredutível. Então, existem subfluxogramas máximos únicos fj e um esquema estruturado S, tal que F = S(f1,f2,...,fm).

Destes teoremas resultam as seguintes proposições:

Proposicão 1

Se F é um fluxograma e i $\in$ F, então existe uma cadeia(série) máxima única de 1 para n+1.

Proposição 2

Seja G = (∇,Δ) um subfluxograma de F. Então, ∇ < Δ numa árvore dominante de F. ∇ e Δ denotam o início e o fim, respectivamente, e < é uma relação de ordem parcial.

Proposição 3

Seja j < k na árvore dominante do fluxograma F. Então, o intervalo I = (j,k) é um subfluxograma de F se e sómente se:

$h \rightarrow i \Rightarrow i = j$

$h \notin I, \quad i \in I$

1: a(2)

2: b(3)

3: A(4,5)

4: c(2)

5: B(7,6)

6: d(8)

7: e(10)

8: C(9,11)

9: f(6)

10: D(2,15)

11: g(12)

12: E(13,14)

13: h(11)

14: F(15,2)

FIG. 4.11(a) - FLUXOGRAMA ESQUEMÁTICO

LEGENDA

+= sim, - = nao

a - I=0, R=' '

b - Ler Tx(I)

A - Tx(I) ja existe

c - Taxa ja existe

B - I>4

d - Introduza o valorM

e - Taxa errada, Tecle "C" para continuar

C - ValorM>1000 ou <2000

f - Valorm errado, Digite novo valor

D - R=C

g - Introduza o valorp

E - ValorM > ValorM

h - ValorP errado, Digite novo valor

F - Trabalho terminado.

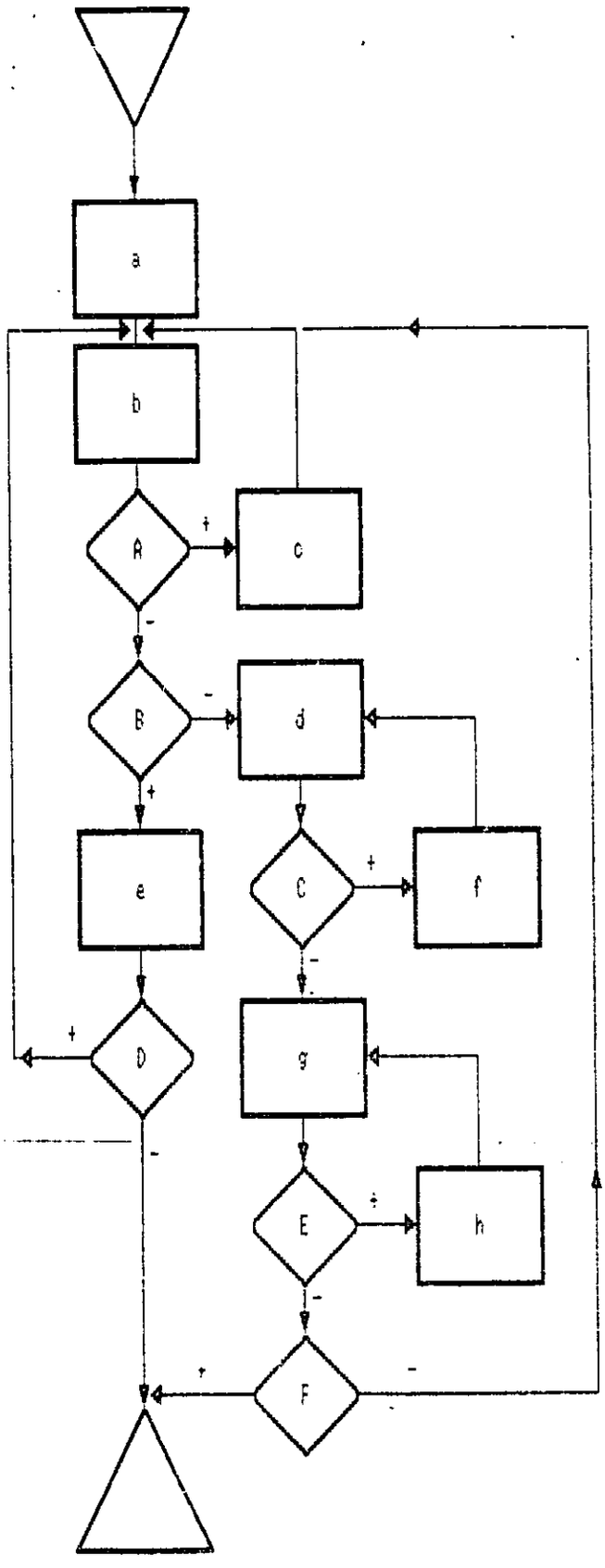


FIG 4.11(b) - FLUXOGRAMA ASSOCIADO

Com estas noções, podemos identificar, do fluxograma anterior, alguns sub-fluxogramas, figuras 4.12(a),(b) e (c).

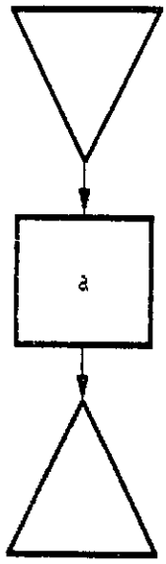


FIG. 4.12(a)
SUBFLUXOGRAMA (1,2)

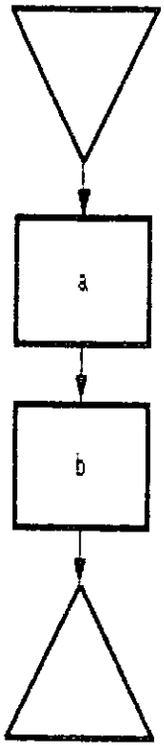


FIG. 4.12(b)
SUBFLUXOGRAMA
(1,3)

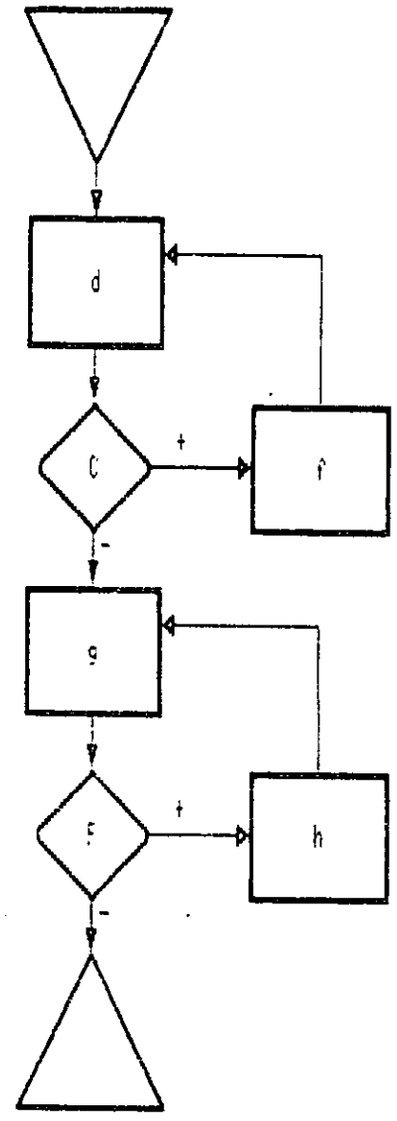


FIG. 4.12(c)
SUBFLUXOGRAMA (6,14)

Com a notação introduzida, denotar-se-á os subfluxogramas acima como (1,2), (1,3) e (6,14), respectivamente.

Finalmente, temos os algorítmos de decompósição dos fluxogramas e da construção da árvore dominante, fig 4.13.

Algorítmo 1

Na terminologia da proposição 1, o sucessor de i numa cadeia máxima única de i para n+1, é chamado "dominador imediato" de i $\in$ F. Este dominador é obtido como segue:

- para i : x(j) o dominador imediato é j;
- para i : P(j,k) todos os caminhos cíclicos livres de i para n+1 podem ser considerados e a primeira junção comum para todos eles é o dominador imediato de i, onde x(j) denota instrução de atribuição e P(j,k) instruções de decisão.

Algorítmo 2

Usando as proposições 2 e 3 e a árvore dominante é necessário encontrar a sub-série apropriada.

$1 = i0 < i1 < \ldots < in = n+1$ da série máxima
$1 = \delta 0 < \delta 1 < \ldots < \delta n = n+1$ na árvore dominante.

Algorítmo 3

Com o uso da árvore dominante e da proposição 2 são apenas considerados os intervalos necessários na árvore dominante e da restante lista apagam-se aqueles que não obedecem as condições da proposição 3 e, finalmente, resta o subfluxograma máximo.

A decomposição dos fluxogramas em subfluxogramas é vantajosa porque facilita a estruturação individual dos módulos além de que permite a divisão do trabalho entre os programadores.

DOMINADOR	IMEDIATO
1	2
2	3
3	5
4	2
5	15
6	8
7	10
8	11
9	6
10	15
11	12
12	14
13	11
14	15

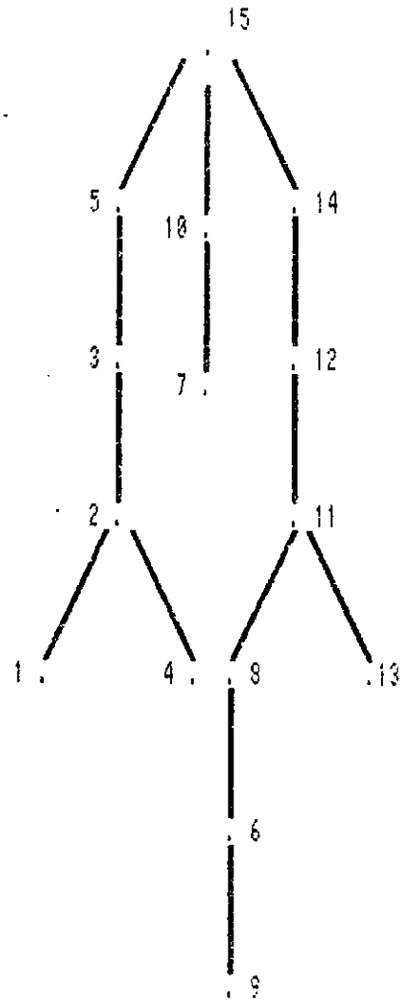


FIG. 4.13 - ÁRVORE DOMINANTE

4.4 - VANTAGENS E DESVANTAGENS

As vantagens dos fluxogramas são:

- Fluxogramas bem desenhados ilustram com maior clareza o fluxo lógico.

- São bons auxiliares de memória, ajudam o analista a gerar um conjunto completo e preciso de instruções de controlo.

- São eficientes quando usados para desenhar algoritmos que se baseiam em decisões com um número de caminhos alternativos não superior a 2 ou 3 [Davis, 1987].

- Resumem muita informação técnica.

- São eficazes para esclarecer elementos do pseudo-codigo.

As desvantagens dos fluxogramas são:

- Uma ilustração de um programa completo (com todos os detalhes) através de fluxograma, torna-se difícil de seguir e impossível de manter.

- Devido aos motivos mencionados no ponto anterior, a elaboração de fluxogramas é um dispêndio de tempo e esforço.

- Utilizam símbolos técnicos de processamento de dados (discos, fita, etc), o que causa embaraços aos usuários estranhos ao assunto.

- Não é útil na "modelagem" do sistema para os usuários.

4.5 CONCLUSÃO

Para além dos que estão aquí ilustrados, existem outros tipos de fluxogramas.

Na etapa do estudo de um sistema, são normalmente usados os fluxogramas de sistema e lógico.

O fluxograma de sistema é usado nas etapas de estudo de viabilidade (fluxograma do sistema existente) e desenho de sistema (fluxograma do sistema proposto), enquanto que o fluxograma de lógica do programa é usado na etapa do desenho detalhado [Davis, 1978].

A diferença principal entre o fluxograma e o DFD, reside no facto de no primeiro se mostrarem fluxos de controle enquanto que no segundo mostram-se fluxos de dados.

Na decomposição do fluxograma, além de se mostrar a lógica do programa, faz-se a análise do próprio fluxograma.

5 - PSEUDOCÓDIGO

O pseudocódigo é um dos métodos para especificação de módulos.

Este conceito muito importante para a fase do desenho do programa tem sido utilizado em várias metodologias e definido de várias maneiras, como a seguir se pode ver.

5.1 DEFINIÇÕES

Para [Grauer e Grawford, 1978], Pseudocódigo é uma forma de expressar o detalhe lógico de um programa.

[Meilir, 1988] e [Neto, 1988] são de opinião que o pseudocódigo é uma linguagem de especificação informal, muito flexível, que é usada na organização dos pensamentos dos programadores antes da codificação. É uma ferramenta de programação estruturada que pode ser usada em projectos estruturados para esclarecer a rotina detalhada de procedimentos de algumas caixas pretas do diagrama de estrutura.

[Radford e Haigh, 1986], afirmam que "pseudocódigo é uma linguagem parecida com uma linguagem de programação real, mas que evita detalhes excessivos, recorrendo de vez em quando ao inglês comum".

5.2 - SUA IMPORTÂNCIA NA FASE PREPARATÓRIA DE IMPLEMENTAÇÃO

Um programa, depois de elaborado, é usado por outras pessoas ou mesmo pelo próprio autor. Tem-se verificado, frequentemente, a necessidade de alterar certas àreas de codificação, sem contudo mudar a linha de pensamento. A base desta modificação são os algorítmos, escritos em pseudocódigo, através dos quais o autor se orienta.

O pseudocódigo, é uma ferramenta do analista e do programador. Esta técnica é composta, fundamentalmente, de regras.

Para demonstrar as regras do pseudocódigo, vamos considerar um exemplo:

Cálculo das despesas mensais com consultores prestando serviços em Moçambique.

Cada consultor é facturado em Escudos e ao dia, de acordo com a sua categoria Txi. Usa-se para o cálculo a fórmula (Txi/30)*n, sendo n o número de dias de trabalho. Para os consultores que vêm por algum período , incluem-se como dias de trabalho os dias de viagem de partida e de regresso de e para Lisboa.

Para além deste montante facturado, cada consultor tem direito a subsídio de alojamento em Dólares e subsídio de alimentação em Meticais. O subsídio de alimentação é retirado sempre que o consultor estiver em gozo de férias. A taxa fixa mensal de subsídio de alimentação é de 300 mil Meticais enquanto que a de alojamento está fixada em 1600 Dólares americanos. [Nhambire e Guambe, 1993].

A estrutura do pseudocódigo, muitas vezes, baseia-se numa linguagem real de programação, como COBOL, FORTRAN ou PASCAL.
Existem muitas versões diferentes. Contudo, a maioria incorpora as três convenções da programação estruturada:

- Sequência, decisão e repetição, [Davis, 1987].

Como se pode observar, estas três convenções foram já abordadas quando nos referimos aos fluxogramas, (cap.4).

Para [Radford e Haigh, 1986], que se baseia na linguagem de programação Pascal, as regras do pseudocódigo são as seguintes:

1. A primeira declaração em cada algorítmo deve ter um nome que marca o seu ponto inicial.

   O nome é escrito em maiúscula e seguido de dois pontos. Ex: DESP_USD:

2. As quantidades de entrada e saída, incluindo nomes para variáveis e constantes, devem ser identificadas e escritas em maiúsculas. Todas as constantes a serem usadas num programa recebem seus valores por meio de um comando que, igualmente, se usa para atribuir valores iniciais a contadores e acumuladores. O comando é:

   ```
   set...to...
   ```

3. O tipo e tamanho das variáveis devem ser declarados.

   Ex:
   ```
   DESP_USD:
       set MES to 30
       declare int CODIGO
       declare real N,FACTURA_USD,TAXA
   ```

4. A entrada e saída de variáveis é expressa de forma genérica, sem referenciar dispositivos específicos de entrada e saída. O formato e:

   ```
   get(TAXA)
   put(TAXA)
   put("Mensagem")
   ```

   O comando get(TAXA) significa que o programa deve receber, de uma fonte externa, um valor para a variável TAXA e o comando put(TAXA) significa exibir o valor da TAXA em algum dispositivo de saída.
   A instrução put é também usada para escrever

mensagens. No exemplo acima, a palavra Mensagem, porque está escrita entre aspas, é exibida no dispositivo de saída exactamente como foi escrita.

5. Uma variável recebe um valor dentro do programa através do comando de atribuição, representado pelo símbolo <- ou =, ex: FACTURA_USD <- TAXA/MES*N.
Este comando, lê-se da seguinte forma: FACTURA_USD recebe o valor da TAXA dividido por MES e multiplicado por N. O comando "end" marca o fim do algoritmo.

6. O fim de todo o programa é marcado por "end".
Tomando-se, como exemplo, o módulo para o cálculo de despesas em Dólares, tem-se :

```
DESP_USD:
 set MES to 30
 declare int CODIGO
 declare real N,FACTURA_USD,TAXA
 put("Entra o código do consultor")
 get(CODIGO)
 put("Entra o valor da taxa")
 get(TAXA)
 put("Entra o número de dias trabalhado")
 get(N)
 FACTURA_USD <- TAXA/MES*N
 PUT("A despesa mensal em Dólares é de ",FACTURA_
           USD,"USD")
 end.
```

7. Na modularização, que é a subdivisão de um programa em subprogramas, a chamada dos subprogramas pelo programa principal é feita através da instrução "call", cujo formato é:

```
call NOME DO SUBPROGRAMA
```

Ex: DESP_MZM:

```
S_ALIM <- V_MES/MES*N
end.
```

e, usando também o exemplo do número anterior, obtem-se o seguinte:

```
DESP_USD:
    put("Entra o código do consultor")
    get(CODIGO)
    put("Entra o valor da taxa")
    get(TAXA)
    put("Entra o número de dias trabalhado")
    get(N)
    FACTURA_USD <- TAXA/MES*N
    PUT("A despesa mensal em prestação de serviço é
        de ",FACTURA_USD,"USD")
    end.


    DESPESA_MES:
    set MES to 30
    set V_MES to 300
    declare int CODIGO
    declare real N,FACTURA_USD,TAXA
    declare real S_ALIM
    put("Este programa calcula as despesas mensais em
          Moçambique")
    call DESP_USD
    call DESP_MZM
    put("A despesa mensal em prestação de serviço é
         de ",FACTURA_USD,"USD")
    put("A despesa mensal em alimentação é de",
        FACTURA_MZM,"MTs")
    end.
```

Aquí, o programa principal DESPESA_MES, chama os subprogramas DESP_USD e DESP_MZM.

8. A repetição de um certo número de actividades é feita através do comando estruturado "loop while". O início e o fim de um loop while são marcados pelas palavras "loop while" e "end loop". O sintaxe é:

```
loop while (condição)
     {
         corpo do loop
      }
end loop
```

Ex:

```
I <- 0
loop while (I>4)
   put(I)
   I <- I+1
end loop
```

A condição da saída do loop é: I>=4

9. Contadores e acumuladores são incrementados através de uma das seguintes expressões.

```
increase VARIÁVEL by CONT
decrease VARIÁVEL by CONT
```

onde, geralmente CONT é igual a (i) para os contadores.

10. A estrutura de decisão tem a seguinte apresentação.

```
if (condição)
  {
     bloco de comandos
   }
```

As chavetas são usadas geralmente em tarefas contendo mais de um comando. Por exemplo no Pascal, os delimitadores "begin" e "end" são necessários porque a

condição verdadeira deve consistir em um único comando. O ponto-e-vírgula marca o fim do comando "if".

Pseudocódigo

```
if (x>y)
   { L <-A
     N <- N+1 }
```

Pascal

```
if (x>y) then begin
    L := A;
    N := N+1
end;
```

11- A estrutura de decisão com alternativa dupla apresenta-se do seguinte modo:

EX:

Pseudocódigo

```
if (condição)
      {..bloco de tarefas verdadeiras..}
else
     {..bloco de tarefas falsas..}
end
```

Pascal

```
if (condição)  then begin
      {..bloco de tarefas verdadeiras..}
end
else begin
     {..bloco de tarefas falsas..}
end;
```

12. A expressão em pseudocódigo da estrutura de decisão com alternativas múltiplas é feita da seguinte maneira:

```
select (CHAVE)
 case 1: []
 case 2: []
 case 3: []
   .
   .
 case n: []
default: []
```

Esta estrutura é um macro da estrutura simples de decisão.

O controle é transferido pelo comando "select", para o bloco "case" que tem o mesmo valor da chave "loop" e caso não houver nenhum bloco "case" com esse valor, o controle passa para o bloco "default".

13. O início e o fim de um "loop repeat" são marcados pelas palavras chaves "loop" e "until" respectivamente. Uma expressão booleana controla a execução do loop. Se a expressão fôr falsa, o corpo do loop é repetido e, no caso contrário, é efectuada a saída do loop.

```
loop
  ...
    conteúdo do loop
  ...
until (Expressão booleana)
```

14. A especificação de loops indexados é estruturado como se segue:

```
loop while I goes from INÍCIO to FIM
   ...
     conteúdo do loop
```

```
    ...
end loop
```

Em Pascal, este loop é chamado loop for.

15. A indicação de módulos definidos pelo utente é feita como se segue:

```
function X:integer
    .
    .
end
procedure Y:
    .
    .
end
```

Para aceder o módulo através do programa principal, faz-se referência ao nome da função em um comando, ou usando o comando "call", conforme se viu na regra 7.

As construções sequência, decisão e repetição são ilustradas nos número 3-7, 10-12 e 8 e 13-14, respectivamente.

5.3 - VANTAGENS E DESVANTAGENS

As vantagens do pseudocódigo são: [Radford e Haigh, 1986]

- É independente de qualquer linguagem de programação.
- É projectado para adaptar-se ao processo mental humano e não ao hardware do computador.
- É geralmente usado como documento associado a programação estruturada
- É menos ambígua
- Representa um meio termo entre os extremos de representação de uma linguagem natural (a que falamos e escrevemos), e uma linguagem de programação
- Para dar maior clareza usa a indentação, [Meilir, 1988]
- Pode ser usado como uma ferramenta "top-down" e como técnica de programação estruturada, especialmente quando a linguagem alvo não suporta estas construções, [Meilir, 1988]
- Reduz a margem de erro dos programadores [Meilir, 1988].

As desvantagens do pseudocódigo são:

- É uma opção menos desejável dada a sua aproximação com a codificação final [Meilir, 1988, p.105].

- Codificação dupla dos programas por parte dos analistas ex-programadores, uma vez em pseudocódigo e outra vez no código real, [Davis, 1987].

- Limita a flexibilidade do programador pois, este quer ser informado sobre o que codificar e não como codificar, [Davis, 1987].

5.4 - CONCLUSÃO

Feita esta breve exposição, conclui-se claramente que as metodologias que usam pseudocódigo contam com um excelente auxílio no desenho de um programa, pois torna flexível a realização de quaisquer mudanças antes e depois de elaborar o programa.

As técnicas estruturadas, conforme se concluiu nos capítulos anteriores, estão ligadas entre sí. Enquanto a técnica dos DFD's mostra o fluxo de dados entre os processos, a técnica HIPO mostra a estrutura hierárquica do programa e descreve as entradas, processos e saídas de cada módulo, (o equivalente a processo no DFD).O fluxograma de lógica do programa, por sua vez, mostra os fluxos lógicos ou seja os fluxos de controle e, finalmente, o pseudocódigo transforma estes fluxos lógicos em uma estrutura "quase-código", que se pode aproximar facilmente a qualquer linguagem de programação, razão pela qual é considerada ferramenta do programador.

6 - CONCLUSÃO GERAL

Terminado o estudo das quatro técnicas, pode-se concluir que se trata de bases fundamentais de análise de processos, sem as quais seria muito difícil edificar e manter um sistema.

O método "Top-Down", cujas técnicas, em parte, aquí desenvolvidas, incorpora ferramentas eficientes que garantem a exactidão e a complementaridade.

Do estudo efectuado conclui-se que no processo de análise de um sistema é fundamental que se examine a adequabilidade ou não das ferramentas disponíveis. Conclui-se igualmente que, embora se possa seguir uma metodologia orientadora, a abordagem do mesmo assunto segundo diferentes perspectiva, ou seja, utilizando diversas ferramentas pode ajudar a melhorar a funcionalidade e a eficiência do sistema, porquanto se obtêm maiores possibilidades de validação do mesmo.

A potencialidade de algumas técnicas tais como DFD's, aumenta substancialmente com o uso de pacotes computarizados. Um exemplo desses pacotes são as ferramentas CASE (Computer-Aided Software Engineering).

Através destas ferramentas a informação contida num dicionário de dados ou ficheiro pode ser adicionada por meio de interfaces, as funções podem ser apagadas ou adicionadas, as estruturas hierárquicas podem ser reconfiguradas, etc. Esta ferramenta oferece muitas facilidades para o tratamento de dados e processos.

Nos DFD's, por exemplo, qualquer modificação num diagrama de nível baixo actualiza interactivamente todos os diagramas de nível mais alto.

Estas ferramentas, devido ao seu elevado custo, são geralmente usadas em sistemas muito complexos, uma vez que, a verificação da consistência de dados é feita através de procedimentos de integridade referencial que são muito eficientes e poupam tempo e esforço ao analista, o que não acontece quando se trabalha com sistemas manuais.

As quatro técnicas estão ordenadas segundo a sua analogia de aplicação. Assim sendo, as técnicas dos DFD's e dos HIPO's podem ser usadas de forma alternativa, uma vez que dão resultado idêntico. A diferença das duas técnicas reside no facto de a primeira ser usada na etapa de análise para representar hierarquias de detalhe, enquanto que a segunda é usada na etapa do desenho do sistema para representar controlos e detalhes.
O mesmo acontece com os fluxogramas de lógica do programas e o pseudocódigo que são geralmente usados de acordo com a dimensão do sistema ou mesmo da preferência e experiência do analista.

Para sistemas de pequena dimensão geralmente se usa o fluxograma de lógica do programa para mostrar os fluxos de controlo, que constituem a base para a programação.

O mesmo não se aconselha quando se trata de sistemas de maiores dimensões pois, os fluxos tornam-se incontroláveis, sendo de recomendar, para estes casos, o uso do pseudocódigo, que facilmente se converte a qualquer linguagem de programação.

Isto prova, mais uma vez, que as metodologias estruturadas podem ser aplicadas para o estudo de qualquer sistema, independentemente da sua complexidade.

REFERÊNCIAS BIBLIOGRÁFICAS

- Ashworth, C. e Goodland, M., 1990, SSADM, A PRACTICAL APPROACH, Inglaterra, McGraw-Hill Book Company Europe

- Barker, R., 1990, CASE*METHOD, USA, Oracle Corporation UK Limited

- Bohm, C. e Jacopini, G., 1966, "Computational linguistics", Communication of the ACM, vol 9, 366-371

- Davis, W. S., 1987, ANÁLISE E PROJECTO DE SISTEMAS "Uma abordagem estruturada", Rio de Janeiro, LTC - Livros Técnicos e Científicos

- DeMarco, T., Fevereiro de 1978, ANÁLISE ESTRUTURADA E ESPECIFICAÇÃO DE SISTEMA, Rio de Janeiro-brazil, Editora Campus

- Gane, C. e Sarson, T., 1979, ANÁLISE ESTRUTURADA DE SISTEMAS, Rio de janeiro, Prentice -Hall

- Grauer, R. e Crawford, 1978, M., COBOL - A Pragmatic Approach, USA,Prentice - Hall

- IBM, 1975, HIPO - A Design Aid and Documentation Technique, IBM corporation Technical Publication/Systems.

- Marques J.A e Guedes P., 1990, FUNDAMENTOS DE SISTEMAS OPERATIVOS, Lisboa, Editorial Presença

- Meilir, P. J., 1988, PROJECTO ESTRUTURADO DE SISTEMAS, São Paulo, Editora McGraw-Hill

- Neto, A. F., Furlan J.D. e Higa W., 1988, ENGENHARIA DA INFORMAÇÃO - Metodologias, técnicas e ferramentas, São Paulo, Editora Mcgraw-Hill

- Nhambire, C.J. e Guambe, A. A., 1993, SISTEMA DE CONTROLO DA FACTURAÇÃO, Maputo

- Prather, R. E. e Giulieri, S.G., 1981, "Decomposition of flowchart schemata",The Computer Journal, vol 24 (3), 258-262

- Radford e Haigh, 1986, TURBO PASCAL para IBM PC e compatíveis,Editora Guanabara, Rio de Janeiro.

- Sommerville.I, 1992, SOFTWARE ENGINEERING, Addison - Wesley