# 4. Modificações na Análise de Sistemas

- É necessário estar ciente das técnicas atuais e das modificações ocorridas com o passar do tempo.
- Há basicamente três motivos para conhecer a evolução da análise de sistemas:
  - Ajuda a perceber a evolução de uma empresa
    - ✓ Mudar de emprego
    - ✓ Sugerir a evolução
    - ✓ Ocupar cargos de liderança para alavancar as mudanças
  - É importante conhecer a abordagem anteriormente adotada pela organização e se há algum tipo de transição em andamento
  - A noção de transição é importante pois a análise de sistemas é dinâmica:
    - ✓ Novas ferramentas
    - ✓ Modificações em ciclos de vida

## 4.1. A passagem para a Análise Estruturada

- Até o final da década de 70, os requisitos dos usuários eram documentados através de uma narrativa no idioma adequado.
- Os primeiros autores sobre Análise Estruturada mostraram que essa forma de especificação padecia de grandes problemas.
  - Monolíticos: Era necessário ler todo o documento para entender. Isso dificultava a compreensão se fosse necessário estudar apenas uma parte.
  - Redundantes: A dificuldade de atualizar e revisar o documento conduz à inconsistência.
  - Ambíguos: usuários, analistas, projetistas e programadores têm interpretações diferentes do documento.
  - Manutenção impossível: A especificação estava obsoleta antes mesmo do final do projeto.
- Como conseqüência, não se tem idéia do que muitos sistemas desenvolvidos nas décadas de 60 e 70 fazem porque os analistas e programadores que os desenvolveram não estão mais presentes.
- Apesar das técnicas de Programação Estruturada e Projeto Estruturado terem sido adotadas, era necessário que houvesse uma evolução na forma de especificar os Requisitos do Usuário.
  - "Poderia se chegar a um desastre com mais rapidez do que nunca".
- A especificação dos requisitos deveria ser:
  - Gráfica
  - Particionada
  - Sem redundância

## 4.2. Modificações na Análise Estruturada

- Alguns anos de experiência prática indicaram que eram necessárias algumas alterações, cujas principais são:
  - Evitar a construção de modelos "físicos" e "lógicos" do sistema atual.
    - Politicamente perigosa (muito tempo gasto com algo que vai ser descontinuado)
  - A distinção vaga entre os modelos lógico e físico (dependente da tecnologia)
    - Modelo lógico => Modelo essencial (essência do sistema)
    - Modelo físico => Modelo de implementação (considera aspectos tecnológicos)
  - Carência de ferramentas de modelagem para construir sistemas de tempo-real
    - Introdução dos Diagramas de Transição de Estado (DTE)
  - Necessidade de modelar as estruturas de dados do sistemas
    - Introdução dos Diagramas de Entidades-Relacionamentos
  - Melhor integração entre as ferramentas (DFD, DER, DTE e DD)
  - Utilização da subdivisão (particionamento) por eventos no lugar do Diagrama de Contexto

## 4.3. Surgimento das Ferramentas Automatizadas de Análise

- Trabalho artístico de criar os diagramas
- O grande problema é dar manutenção nos diagramas
  - Muitas modificações durante a análise
  - Grande quantidade de diagramas
- Dificultou a aceitação da Análise Estruturada Trabalho artístico de criar os diagramas
  - Analista preferia deixar o diagrama desatualizado
  - Analista não subdividia o modelo do sistema em modelos de nível mais baixo
  - Os Projetistas e Programadores não mantinham os diagramas atualizados durante a implementação
- Não havia verificação automática de consistência nos diagramas (era necessário fazer inspeções visuais)
- Custo elevado das ferramentas e dos terminais gráficos

### *4.4. Uso da Prototipação*

- Surgimento de ferramentas de Prototipação
- A Análise Estruturada levava muito tempo:
  - Modelagem do sistema novo só começa após a do sistema atual

- Como os Diagramas não geravam código, suspeitava-se que o tempo gasto na implementação seria igual
- Os primeiros projetos levavam mais tempo pois os Analistas não estavam acostumados com as técnicas
- muito de programação seria o mesmo se não fosse feita análise

- A prototipação se concentra na definição da interface homem-máquina
- Evita os detalhes que são capturados através da Análise e do Projeto

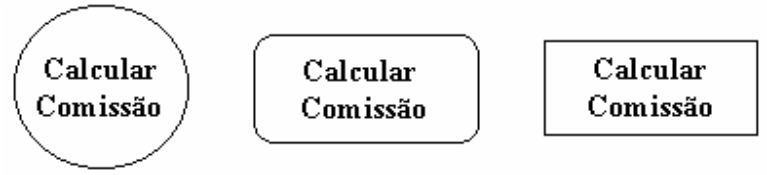
## 4.5. Diagrama de Fluxo de Dados

- Principal ferramenta de modelagem funcional
- Modela o sistema como uma rede de processos funcionais, interligados por dutos e tanques de armazenamento
- Pode ser usado para descrever processos computadorizados e não computadorizados
- Também chamado de DFD (abreviatura), Diagrama de Bolhas, Modelo de Processo, Diagrama de Fluxo de Trabalho e Modelo Funcional
- Um DFD é composto de Processos, Fluxos de Dados, Depósitos de Dados e Entidades Externas

### 4.5.1. Componentes de um DFD

**Processos**

- Também conhecido como bolha, função ou transformação
- Representam transformações de fluxo(s) de dados de entrada em fluxo(s) de dados de saída
- O nome do processo deve descrever o que ele faz
- Geralmente provoca mudanças de estrutura, conteúdo ou estado
- Representações gráficas possíveis:



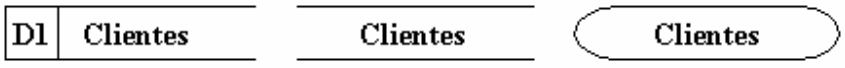

**Fluxos de Dados**

- Representam caminhos por onde passam os dados
- São representados através de setas que indicam o destino do dado
- Têm nomes que devem constar no dicionário de dados

- Um mesmo fragmento de dados pode ter significados diferentes em pontos distintos de um DFD (CPF-Válido e CPF-Inválido)
- Um fluxo apenas não modifica os dados durante o transporte
- Transportam dados entre os elementos do DFD
  - Processo ⇔ Processo
  - Entidade Externa ⇔ Processo
  - Depósito de Dados ⇔ Processo
- Tipos de fluxos
  - Fluxo externo: entre Entidade Externa e Processo
  - Fluxo interno: entre dois Processos
  - Fluxo de acesso à memória: entre Processo e Depósito
  - Fluxo de erro ou rejeição: para fora de um Processo
- Nomenclatura:
  - Cada fluxo deve ter um único nome
  - O nome deve identificar os dados transportados pelo fluxo
  - Exemplos: Dados-Fatura, Recibo-Pagamento, Dados-Cliente

**Depósitos de Dados**

- Representa uma coleção de pacotes de dados em repouso
- Nem sempre um depósito de dados é um arquivo ou SGBD. Pode representar microfilmes, pastas de arquivos em papel e diversas outras formas não computadorizadas
- Representações gráficas de um depósito de dados:

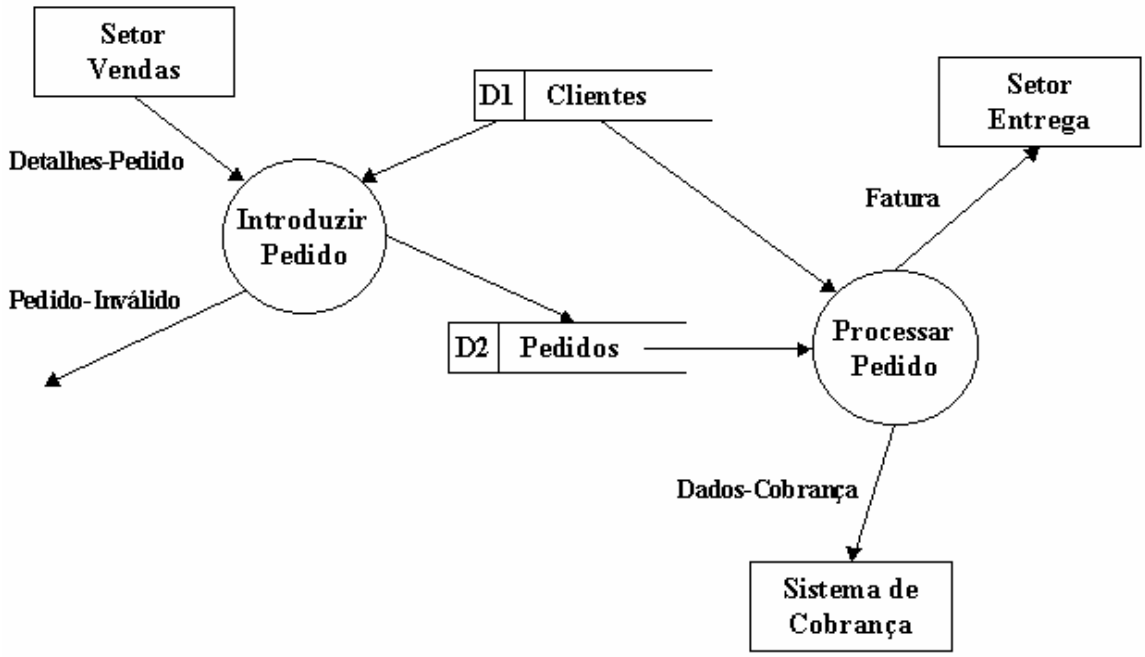



- Quando um pacote de dados é recuperado (ou inserido) por completo do depósito de dados, pode-se omitir o rótulo do fluxo
- Nomenclatura:
  - Deve estar no plural
  - Pode receber o nome do fluxo de dados (no plural)

**Entidades Externas**

- Também chamados de Terminadores
- São as fontes/destinatários das informações que entram/saem do sistema
- Os procedimentos executados pelas entidades externas não são especificados no modelo por não fazerem parte do sistema
- Normalmente é uma pessoa, um grupo de pessoas, uma organização externa, um setor dentro de uma empresa
- Pode representar um outro sistema
- Representação gráfica de uma Entidade Externa:


- Nomenclatura:
  - No plural quando se referir a um grupo de pessoas (Clientes)
  - Deve conter o nome do setor ou organização externa (Diretoria de Negócios)
  - Deve ser incluída a palavra sistema quando se tratar de um sistema (Sistema de Contabilidade)




### 4.5.2. Diretrizes para elaboração de DFD

- Existem algumas diretrizes que auxiliam a criar DFD's com sucesso, ou seja, evitam a criação de:
  - DFD's incorretos (incompletos ou logicamente inconsistentes)
  - DFD's agradáveis (facilmente examinados pelo usuário)

**Escolher Nomes Significativos**

- Evitar nomes para processos como: Fazer serviço, Manipular entrada, Cuidar dos clientes e Processar dados

**Deve-se numerar Processos**

- A numeração basicamente duas utilidades:
  - Permitir localizar os processos no diagrama facilmente
  - Facilita a identificação, a partir dos digramas mais detalhados, do processo foi explodido

- Não importa a maneira desde que seja consistente
- A numeração não indica seqüência pois o DFD é uma rede de processos assíncronos que se intercomunicam

**Evitar DFD Complexos**

- Evitar colocar elementos demais no digrama
- Deve caber facilmente em uma página
- O DFD deve modelar corretamente as funções que um sistema deve executar e as interações entre elas.
- Deve ser lido e entendido facilmente pelos usuários

**Refazer tantas vezes quantas forem necessárias**

- Um DFD deve ser refeito até que:
  - Esteja tecnicamente correto
  - Aceitável pelo usuário
  - O Analista não tenha vergonha de apresentá-lo à diretoria.

**Criar diagramas esteticamente agradáveis**

- Manter consistentes o tamanho e a forma das bolhas
- Fluxo de dados cursos versus retos (questão de gosto)
- Diagramas desenhados à mão versus gerados por máquina
  - Os desenhados à mão passam a sensação de que ainda podem ser modificados
  - Os gerados por máquina são mais limpos

**Certificar-se de que o DFD seja logicamente consistente**

- Evitar "poços sem fundo" (processos que só recebem entradas)
- Evitar processos com geração espontânea (processos que não recebem entrada mas produzem saídas)
- Cuidado com fluxos e processos sem rótulos
- Cuidado com depósitos que tenham somente leitura ou escrita

**Posição dos elementos**

- O processo origem deve ficar à esquerda ou acima do processo destino
- As entidades externas devem ser desenhadas nas bordas do desenho:
  - As de entrada, à esquerda ou acima
  - As de saída, à direita ou abaixo
- Os depósitos de dados devem ser distribuídos no meio do desenho, entre os processos

**Duplicação de elementos**

- Pode-se duplicar Entidades e Depósitos para evitar cruzamento de fluxos e melhorar a organização do diagrama
- Um mesmo fluxo de dados pode aparecer mais de uma vez no mesmo DFD
- Não faz sentido duplicar processos

### 4.5.3. DFD com Níveis

- O DFD de sistemas não triviais é muito complexo
- Para evitar que tudo seja definido em um único diagrama (difícil de ser entendido e mantido), criam-se DFD's que detalham um processo de um nível mais alto

**Diagrama de Contexto**

- É o DFD de nível mais alto
- Dá a visão das principais funções do sistema
- Contém um processo (representa o sistema), os fluxos externos e as entidades externas

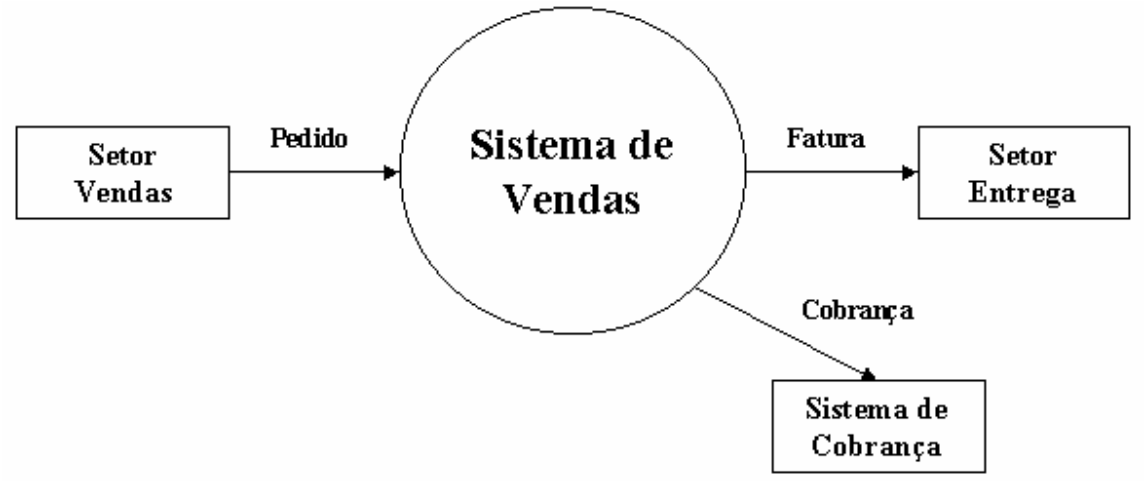



**Diagrama Nível 0**

- É o primeiro detalhamento do diagrama de contexto
- Contém as macro-funções do sistema

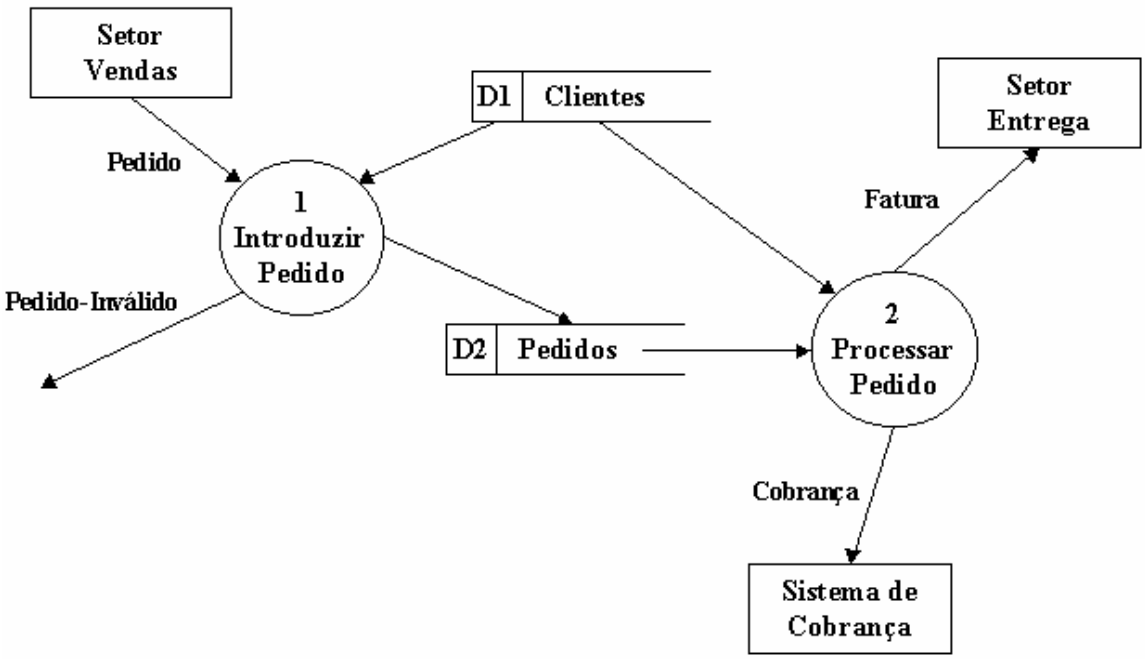
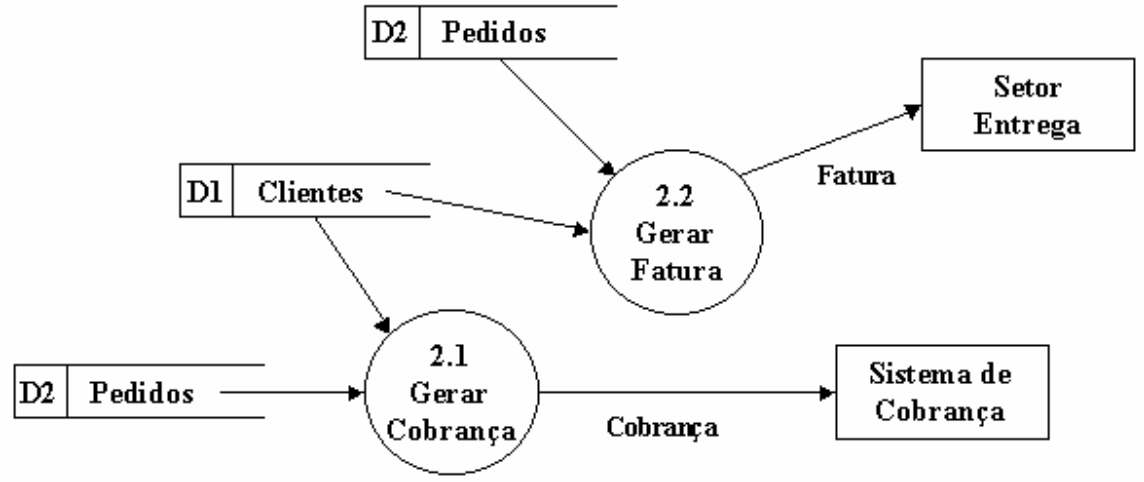



**Diagrama de Níveis Intermediários**

- São os diagramas que mostram a decomposição (detalhamento ou explosão) de cada processo de nível mais alto
- A quantidade de níveis depende de fatores como complexidade e porte do sistema
- Em geral, a decomposição deve terminar quando for possível especificar processo em uma página



## 4.6. Dicionário de Dados

- É necessário descrever a composição dos dados de alguma forma.
  - A forma narrativa é longa e sujeita a erros
  - É necessário usar uma notação compacta e concisa
- **Elementos de dados** são dados que não necessitam de decomposição
- **Estrutura de dados** são composições de elementos de dados e/ou de outras estruturas de dados
- A definição no DD é feita de forma *Top Down*
- O dicionário de dados define os elementos de dados descrevendo:
  - O significado de **fluxos** e **depósitos**
  - A composição de pacotes agregados de dados que se movimentam pelos **fluxos** (Ex: Endereço pode ser divido em itens elementares como cidade, estado etc.)
  - A composição dos pacotes de dados nos **depósitos**
  - Os valores e unidades relevantes de partes elementares de informações dos **fluxos** e **depósitos**
  - Os detalhes dos **relacionamentos** entre os **depósitos** realçados em um DER

### 4.6.1. Notação

- Há vários esquemas de notação. Porém, o mais comum é o seguinte:

| | |
|---|---|
| = | É composto de |
| + | E (concatenação) |
| ( ) | Opcional |
| { } | Iteração |
| [ ] | Escolha de uma das opções alternativas |
| * | Delimitador de comentário |
| @ | Identificador (campo chave) de um depósito |
| \| | Separa opções alternativas na construção [ ] |

**Exemplo:** definição de um nome (estrutura de dados)

```
nome =                  * Nome completo do cliente *
                        título-cortesia + primeiro-nome + (nome-
                    intermediário) + último-nome
título-cortesia =       [Sr.|Srta.|Sra.|Sras.|Dr.|Professor]
primeiro-nome =         {caractere-válido}
nome-intermediário =    {caractere-válido}
último-nome =           {caractere-válido}
caractere-válido =      [A-Z|a-z|0-9|'| ]
```

### 4.6.2. Definições

- Uma definição de um item de dados é apresentada com o símbolo "=", que deve ser lido como "é definido como", ou "é composto de", ou simplesmente "significa"
- A notação **A = B + C**, significa A é composto de B e C
- O **significado** do dado no contexto da aplicação deve ser colocado na forma de **comentário**

### 4.6.3. Elementos opcionais

- Um elemento de dados é opcional quando sua presença no elemento de dados composto não é obrigatória

**Exemplo:** um cliente deve ter um endereço e pode informar um endereço de remessa

```
Cliente =       Endereço + (Endereço-Remessa)
```

### 4.6.4. Iteração

- Usado para indicar a ocorrência repetida de um componente de um elemento de dados

**Exemplo 1:** um pedido que é composto de um nome do cliente, um endereço de remessa e zero ou mais itens

```
Pedido =        Nome-do-Cliente + Endereço-Remessa + {Item}
```

**Exemplo 2:** um pedido que é composto de um nome do cliente, um endereço de remessa e de 1 a 10 itens

```
Pedido =          Nome-do-Cliente + Endereço-Remessa + 1{Item}10
```

**Exemplo 3:** um pedido que é composto de um nome do cliente, um endereço de remessa e pelo menos um item

```
Pedido=           Nome-do-Cliente + Endereço-Remessa + 1{Item}
```

**Exemplo 4:** um pedido que é composto de um nome do cliente, um endereço de remessa e no máximo 10 itens

```
Pedido =          Nome-do-Cliente + Endereço-Remessa + {Item}10
```

### 4.6.5. Seleção

- Indica que deve ser selecionada uma das opções apresentadas

**Exemplo:** definindo o estado civil

```
Estado-Civil =          [Solteiro | Casado | Divorciado | Separado | Outro]
```

### 4.6.6. Sinônimo

- É necessário quando os usuários usam termos diferentes para um mesmo dado

**Exemplo:**

```
Número-do-Item =          1{Dígito}5
Número-da-Peça =          * Sinônimo de Número do Item *
Dígito =                  [0 | 1 | 2 | 3]
```

### 4.6.7. Definição de Depósitos

- A definição deve vir entre {} para indicar a existência de 0 a n ocorrências
- Coloca-se o caractere @ antes do item de dado que identifica uma ocorrência(instância) do depósito

**Exemplo:** definindo depósitos de Clientes e Funcionários

```
Clientes =      { @CPF-CNPJ + Nome + Data-cadastro + Endereço }
Funcionários =  { @Matrícula + Nome + Data-contratação + Endereço +
                  {@Telefone + Descrição} + {@RG-Dependente + Nome} }
```

***11.08.2004***

| ***Nome*** | ***Assinatura*** | ***E-Mail*** |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |