

## **Guia de Estudo - Parte II**

### **Programa da Parte II**

- 2. Modelagem de Software
  - 2.1 Software
    - 2.1.1 Conceitos
    - 2.1.2 Tipos de Software
  - 2.2 Modelagem de Software
    - 2.2.1 Conceitos
    - 2.2.2 Modelagem Bottom-Up e Top-Down
  - 2.3 Ciclo de Vida de Sistemas
    - 2.3.1 Conceitos
    - 2.3.2 Tipos de Ciclos de Vida de Sistemas
  - 2.4 Ferramentas de Modelagem
    - 2.4.1 Diagrama de Fluxo de Dados (DFD)
    - 2.4.2 Diagrama Entidade-Relacionamento (DER)
    - 2.4.3 Diagrama de Transição de Estados (DTE)
    - 2.4.4 Estudo de Casos de Modelagem

### **Bibliografia**

[POM02] POMPILHO, S. Análise Essencial – guia prático de Análise de Sistemas. Rio de Janeiro, Editora Ciência Moderna, 2002.

### **Bibliografia Detalhada ( segundo a ordem dos tópicos do programa )**

- [POM02] Capítulo 3, pg. 21 a 26
- [POM02] Capítulo 4, pg. 27 a 32
- [POM02] Capítulo 7, pg. 45 a 49
- [POM02] Capítulo 8, pg. 61 a 70
- [POM02] Capítulo 10, pg. 77 a 90
- [POM02] Capítulo 16, pg. 179 a 190

## Sistemas Baseados em Computador

Os Sistemas Baseados em Computador são um conjunto ou arranjo de elementos que são organizados para atingir alguma meta pré-definida pelo processamento de informação. São formados por 6 elementos: Hardware, Software, Pessoal, Base de Dados, Documentação e Procedimentos.

1. **Hardware:** Dispositivos eletrônicos que fornecem a capacidade computacional, dispositivos de interconectividade que possibilitam o fluxo de dados e dispositivos eletromecânico que interagem com o mundo externo.
2. **Software:** Programas de computador, estruturas de dados e documentação correlacionados que servem para realizar o método lógico, procedimento ou controle necessário.
3. **Pessoal:** Usuários e Operadores do Hardware e Software.
4. **Base de Dados:** Coleção organizada de informações que se tem acesso através do Software.
5. **Documentação:** Informação descritiva que mostram o uso e/ou operação do sistema.
6. **Procedimentos:** Passos que definem o uso específico de cada elemento do sistema ou o contexto de procedimento no qual o sistema reside.

## Software

Software são (1) instruções (programas de computador) que quando executados fornecem a função e desempenho desejados, (2) estruturas de dados que permitem aos programas manipularem adequadamente a informação e (3) documentos que descrevem adequadamente a operação e uso dos programas.

O Software assume um papel duplo, sendo produto e ao mesmo tempo o veículo para entrega de um dos mais importantes produtos de nossa época – a informação. Do ponto de vista de Analista de Sistemas o produto é um conjunto de programas, dados e documentos, mas do ponto de vista do Usuário o produto é a Informação resultante do processamento do Software. O software transforma dados de modo que possam ser mais úteis em determinado contexto; gera informação comercial para aumentar a produtividade; fornece portais para a rede de informações mundial (Internet), proporcionando meios de obter informações em todas as suas formas.

## Características do Software

O Software é construído por um processo humano criativo (Análise, Projeto, etc) traduzido na forma de um programa gravado em uma mídia física. O software entretanto é um sistema lógico e não um sistema físico tendo características próprias que são consideravelmente diferentes daquelas do Hardware.

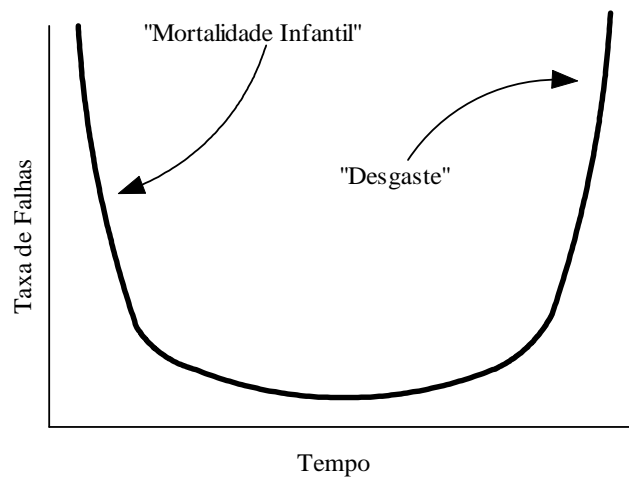
1. O Software é desenvolvido, ou passado por um processo de Engenharia, mas não é manufaturado no sentido clássico.

Apesar das semelhanças entre o desenvolvimento do Software e a fabricação do Hardware, são duas atividades fundamentalmente diferentes. Em ambas a alta qualidade é conseguida com um bom projeto, mas a fase de fabricação do Hardware pode gerar problemas de qualidade que são inexistentes, ou facilmente corrigidos, para o Software. Os custos do Software são concentrados na fase Engenharia, o que significa que os projetos de Software não podem ser geridos como projetos de fabricação.

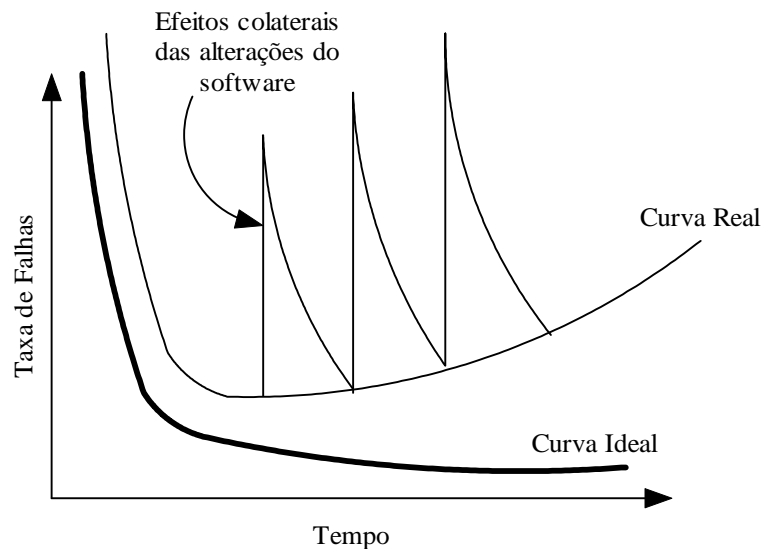
2. Software não de “desgasta”

A primeira figura abaixo mostra a taxa de falhas com função do tempo para o Hardware. A curva mostra que o Hardware exhibe taxas de falhas relativamente altas no início da vida, atribuídas a defeitos no projeto ou na fabricação, depois os defeitos são corrigidos e as taxas caem para um nível constante. Com o passar do tempo, entretanto, as falhas começam a aumentar devido ao desgaste.

A segunda figura mostra a uma curva “ideal” e uma curva “real” para o Software. O Software não seria suscetível a “desgaste”, portanto deveriam ter uma queda constante da taxa de falhas. Isto porém não é verdade pois apesar do software não sofrer “desgaste” ele se “deteriora”. Isto ocorre em função das falhas introduzidas durante as manutenções que serão executadas, pois a medida que manutenções, que alteram o Software original, forem sendo feitas novos defeitos são introduzidos. Outro problema é que a manutenção do Hardware envolve normalmente peças sobressalentes, o que não ocorre com o software, pois defeitos de software normalmente são defeitos de projeto e devem ser corrigidos.



### Curva de Falhas do Hardware



### Curva de Falhas do Software

3. Apesar de a indústria estar se movendo em direção a montagem baseada em componentes, a maior parte do software continua a ser construída sob encomenda.

O Hardware é construído com base em componentes pré-fabricados que são reunidos de uma forma específica para determinado projeto. Os componentes foram criados de forma que os Engenheiros pudessem se concentrar nos elementos realmente inovadores de um projeto em não em reinventar componentes que já existem.

A reutilização de componentes é algo natural nas Engenharias, mas ainda é um conceito novo no Software. Um componente de software deve ser projetado de forma que possa ser reutilizado em muitos programas diferentes. Na década de 60 surgiram as bibliotecas de sub-rotinas científicas, com implementações de algoritmos conhecidos, que eram reutilizadas em uma ampla gama de programas. Hoje a reutilização de software abrange não apenas estruturas de código, mas também estruturas de dados. Um exemplo são os componentes visuais das interfaces gráficas, como botões, grids e outros.

## **Tipos de Software**

O Software pode ser aplicado em qualquer situação onde um conjunto previamente especificado de procedimentos (algoritmo) tenha sido definido (Uma exceção a esta regra são Sistemas Especialistas e Sistemas de Redes Neurais da área de Inteligência Artificial). A classificação do software é algo complexo em função de sua diversidade, podendo ser separado em algumas categorias.

### 1. Software de Sistemas

Software de Sistemas é uma coleção de programas escrita para servir outros programas. São caracterizados por uma interação intensa com o Hardware do computador, utilização por múltiplos usuários, operação concorrente, compartilhamento de recursos, gestão de processo sofisticada e estruturas de dados complexas.

Exemplos são os compiladores, os editores de programas, os utilitários de manipulação de arquivos, os componentes de sistema operacional, os "drivers" e os programas de conectividade.

### 2. Software de Tempo Real

Software de Tempo real monitora, analisa e controla eventos do mundo real a medida que eles ocorrem. Eles possuem 4 elementos: um que coleta, formata e armazena os eventos, um que analisa, um que gera a saída e um que controla todos os outros. O tempo de resposta varia de 1 milissegundo a 1 segundo, devendo ser compatível com a velocidade dos eventos do mundo real.

Exemplos são sistemas de controle de produção de fábricas e sistemas de controle de tráfego aéreo.

### 3. Software Comercial

Software Comercial é voltado para o gerenciamento de informações empresariais, sendo a maior área de aplicação de Software. Os sistemas "discretos" de Folha de Pagamento, Contabilidade, Controle de Clientes ou

Fluxo de Caixa, evoluíram para Sistemas Integrados de Gestão. Aplicações nesta área processam os dados existentes de modo a facilitar operações comerciais ou a tomada de decisão de gestão de negócios.

Exemplos são Sistemas Integrados de Gestão, Sistemas de Ponto de Venda e Sistema de Informações Gerenciais.

#### 4. Software Científico e de Engenharia

Software Científico e de Engenharia é voltado para o processamento de grandes quantidades de dados.

Exemplos são softwares de cálculo de Engenharia, modelos de sistemas biológicos e cálculo de rotas de satélites.

#### 5. Software Embutido

Software Embutido está gravado dentro de produtos ou sistemas industriais controlando seu funcionamento. Este tipo de software executa funções muito particulares e usualmente bastante limitadas.

Exemplos são software de Telefone Celular, Forno de Micro-Ondas e Injeção Eletrônica de Combustível de automóveis.

#### 6. Software para Computadores Pessoais

Software para Computadores Pessoais são ferramentas de produtividade utilizadas na estação de trabalho da pessoa.

Exemplos são os Editores de Texto, Planilhas de Cálculo, Softwares de Apresentação e de Desenho.

#### 7. Software para Web

Software para Web surgiu com a explosão da Internet, que permitiu criar aplicações ricas em recursos gráficos acessadas remotamente via Internet. Com estes Softwares a Internet se torna um grande computador, que fornece um recurso quase ilimitado de Software.

#### 8. Software para Inteligência Artificial

Software para Inteligência Artificial faz uso de algoritmos não numéricos para resolver problemas complexos que não são passíveis de computação ou análise direta.

Exemplos são Sistemas Especialistas, Sistemas de Reconhecimento de Padrões e Redes Neurais.

## Modelagem de Software

Modelo é uma representação simplificada de alguma parte da realidade. Sua necessidade vem da nossa incapacidade de manipular a realidade diretamente com todas as suas incertezas.

Modelagem significa capturar aspectos importantes do que está sendo modelado, de um certo ponto de vista, simplificando ou omitindo o resto. Os principais objetivos da modelagem são:

- Definir processos, servindo às necessidades da visão que está sendo considerada;
- Representar o comportamento dos processos e os pressupostos nos quais o comportamento está baseado;
- Definir explicitamente Entradas e Saídas do Sistema.

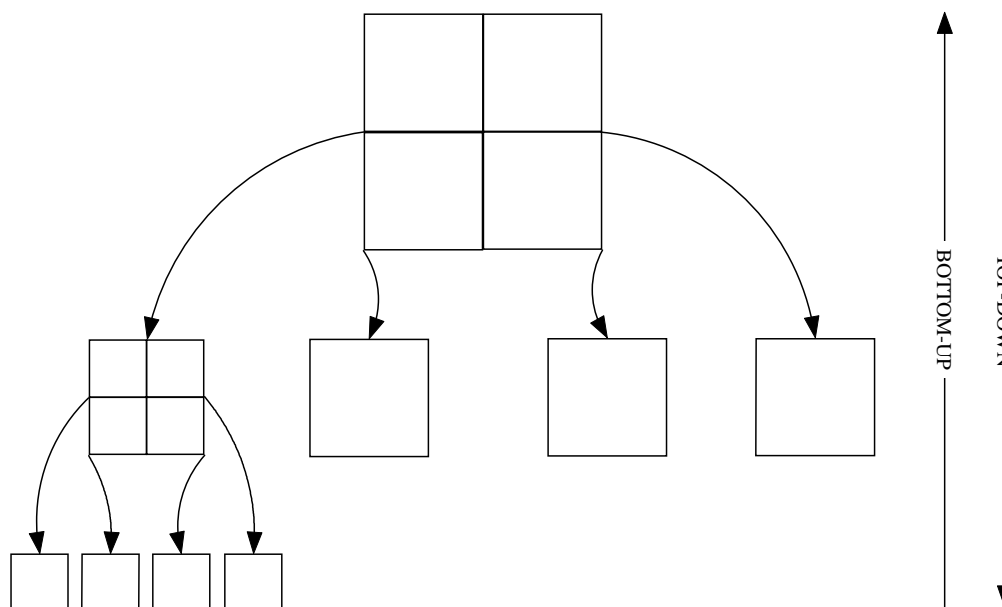
## Fatores de Modelagem

A construção de modelos considera uma série de Fatores Restritivos que são introduzidos para viabilizar a construção do Sistema:

1. **Pressupostos** (de Entradas) reduzem a quantidade de permutações e variações possíveis permitindo assim a um modelo refletir o problema de um modo razoável. Os pressupostos reduzem a quantidade de variações possíveis nas Entradas e no Processamento. Exemplo: Sistema de Pedidos que aceita entrada via digitação e não via Web ou via Palmtop.
2. **Simplificações** (de Processo) permitem que o modelo seja criado no prazo adequado. As simplificações buscam eliminar diferenças entre Processos diferentes simplificando o modelo. Exemplo: Software de Desenho de 2 dimensões (2D) e não 3 dimensões (3D).
3. **Limitações** (de Fronteira) ajudam a delimitar o Sistema. As limitações reduzem a abrangência do Sistema. Exemplo: Sistema de Controle de Aviões para até 2 motores e não 3 ou 4.
4. **Restrições** (de Projeto) vão guiar a forma pelo qual o modelo é criado e a forma pelo qual o modelo é implementado. Exemplo: Sistema de Desenho Auxiliado por Computador (CAD) para ser executado em computadores pessoais (de baixo desempenho) e não em estações de trabalho (de alto desempenho).
5. **Preferências** (de Implantação) que indicam a estrutura preferida para dados, funções e tecnologia. Exemplo: Utilizar Sistema Operacional Linux e não Sistema Operacional Windows.

## Modelagem Top-Down e Bottom-Up

Segundo a Teoria Geral de Sistemas, um Sistema é composto de Entrada, Saída, Processamento e Retroação. A TGS coloca ainda que cada Sistema é dividido em Sub-Sistemas com os mesmos elementos., Assim, um Sistema pode ser visto como uma "árvore" de Sistemas, como mostrado no diagrama abaixo. Cada Sub-Sistema tem os 4 elementos de um Sistema.



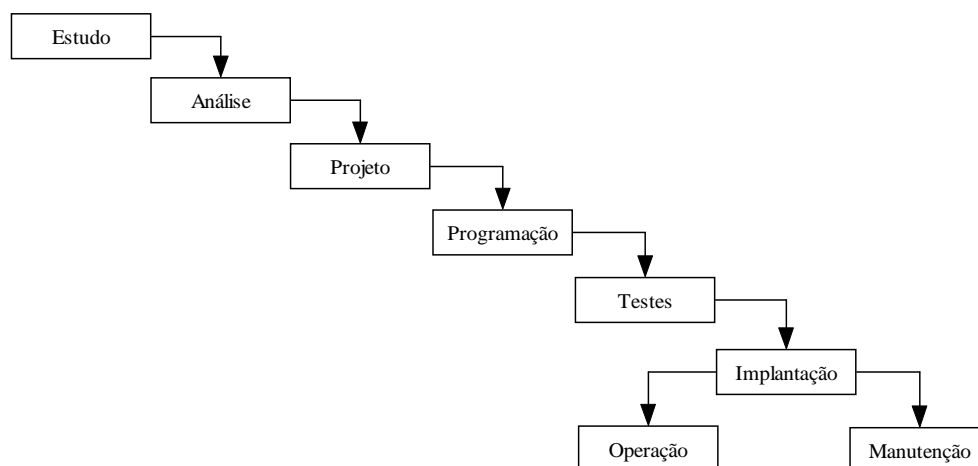
Existem 2 formas de abordar o Sistema: Bottom-Up e Top-Down. A abordagem Bottom-Up não considera a os princípios da TGS, mas os que a antecederam. Ela acredita que para compreender um Sistema devemos compreender as suas partes, ou seja, o todo é a soma das partes. A abordagem Top-Down considera a TGS e assim os Sub-Sistemas devem ser vistos como partes de um Sistema maior, ou seja, o todo é mais do que a soma das partes.

Fazendo uma analogia com a construção de uma Casa, poderíamos dizer que a abordagem Bottom-Up é a abordagem de Pedreiro e a abordagem Top-Down é a abordagem do Engenheiro. Se o Pedreiro for construir uma casa, ele vai começar fazendo as paredes, depois o telhado, depois instalações hidráulicas e elétricas e depois acabamento, e se tudo der certo, ao final teremos uma casa. Se for uma Casa de tamanho médio é provável que seja bem sucedido, mas talvez não a construa da forma mais eficiente. Se o Engenheiro for construir uma casa vai começar com um Projeto Civil/Arquitetônico, Elétrico e Hidráulico, se possível, fazendo uma maquete real ou em computador para mostrar a Casa ao cliente antes de construir. Além disso, ao levantar uma parede ele vai saber que esta parede também terá instalações elétricas e hidráulicas e assim pode prever isso já ao levantar a parede. Imaginemos agora estas 2 visões para construir uma linha completa de Metrô ! A visão do Pedreiro simplesmente não funcionaria, ou então levaria anos a mais a um custo exorbitante para corrigir os erros.

As mesmas colocações valem para a construção de um Software. Se ele for pequeno, é provável que uma visão Bottom-Up funcione, mas em qualquer Software de tamanho razoável para dar resultados interessantes, a abordagem Top-Down deve ser usada. Uma das grandes vantagens da abordagem Top-Down é a chamada "Administração da Complexidade", ou seja, podemos usar a idéia de Caixa-Preta para determinado Sub-Sistemas que ainda não queremos ou não podemos detalhar e assim apenas definir as suas Entradas, Saídas e o Processamento desejado, sem detalhar como este Processamento será obtido.

## Ciclo de Vida de Sistemas

A Engenharia de Software é a área do conhecimento responsável pela criação de Sistemas Baseados em Computador. Existem várias metodologias para desenvolvimento de sistemas, mas praticamente todas compartilham as etapas mostradas no Ciclo de Vida abaixo, principalmente as etapas de Análise, Projeto e Programação (ou Implementação).



**Estudo** se refere ao levantamento inicial e, normalmente, informal do Sistema, as necessidades envolvidas e as possíveis soluções tecnológicas.

**Análise** determina quais são os requisitos do Sistema, respondendo a pergunta "O QUÊ?". A Análise tem como objetivo interpretar e definir a estrutura de um problema que ainda não estava estruturado.

**Projeto** determina como o Sistema vai funcionar, respondendo a pergunta "COMO?". O Projeto tem como objetivo especificar as soluções que serão adotadas para atender a especificação da fase de Análise.

**Programação** ou **Implementação** criar os códigos e estruturas de dados do Sistema, implementando as especificações de Projeto. Testes são necessários para validar o resultado da Programação.

**Implantação** trata de colocar o Sistema em produção, gerando a partir daí processos de **Operação**, que cuidam de manter o Sistema produzindo, e processos de **Manutenção**, que cuidam de corrigir defeitos do Sistema.

O Ciclo de Vida mostrada no figura acima é conhecido como “Ciclo de Vida em Cascata”, pois considera que cada etapa se encerra antes de iniciar a próxima. Esta abordagem traz sérios problemas, pois erros que ocorreram, por exemplo, nas etapas de Análise ou Projeto, não podem ser mais discutidos na etapa de Programação. Uma dos maiores avanços no Ciclo de Vida de Sistema na última década foi o conceito de “Ciclo de Vida Interativo”, ou seja, completa-se um Ciclo Análise-Projeto-Programação, são avaliados os problemas e se inicia outro ciclo Análise-Projeto-Programação. Com isso se faz um desenvolvimento incremental do Sistema o que permite melhorar o Sistema de forma recursiva.

### Análise de Sistemas

A Análise de Sistemas deve tratar, considerando a TGS dos elementos de Sistemas: Entradas, Saídas, Processo e Retroação. Existe ainda o conceito de Eventos que são solicitações que o Sistema recebe e que estão associados a determinadas Entradas que o Sistema deve processar. As Entradas e Saídas são os “Dados”, o Processo são as “Funções” e os Eventos associados a Retroação são o “Controle” do Sistema. Estes 3 elementos estão presentes na história da evolução da Análise de Sistemas definindo as fases pelas quais ela passou.

A Análise de Sistemas Convencional (Não Orientada a Objetos) pode ser dividida em 3 etapas mostradas na tabela abaixo. A Análise “Antiga” foi desenvolvida na década de 70, a Análise Estruturada foi desenvolvida no início da década de 80 e a Análise Estrutura Moderna foi desenvolvida no final da década de 80.

TÉCNICAS	ABORDAGENS	FERRAMENTAS
Análise “Antiga”	Funcional	Textos Fluxogramas
Análise Estruturada	Funcional Dados	Diagrama de Fluxo de Dados Diagrama de Estrutura de Dados Especificações Dicionário de Dados
Análise Estruturada Moderna ou Análise Essencial	Funcional Dados Controle (Eventos)	Diagrama de Fluxo de Dados Diagrama Entidade-Relacionamento Diagrama de Transição de Estados Especificações Dicionário de Dados

### Diagrama de Fluxo de Dados (DFD)

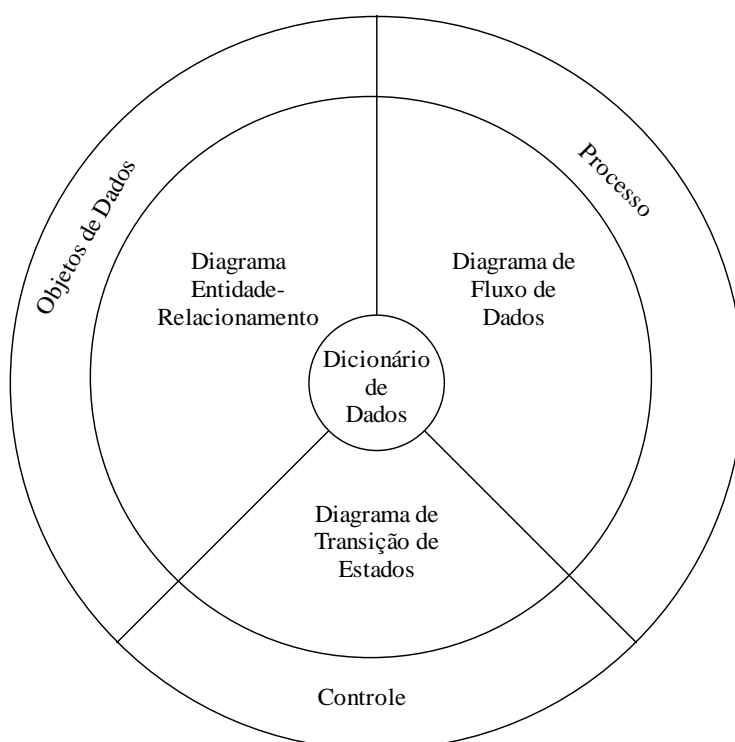
O DFD representa as funções (processo) do sistema, mostrando as relações e o fluxo de dados em ter elas.

### Diagrama Entidade-Relacionamento (DER)

O DER representa as entidades e seus atributos, e o relacionamento entre elas.

### Diagrama de Transição de Estados (DTE)

O DTE representa os estados de um Sistema e a transição entre eles, bem como as condições para que elas ocorram.



O esquema acima mostra o Modelo de Análise de Sistemas com seus 3 diagramas e suas relações. Ao centro está um Dicionário de Dados com todas as definições do Sistema, definições estas usadas nos diagramas. O DER especifica os Dados, o DFD especifica as Funções do Processo e a DTE especifica o Controle.