Bibliografia:

FALBO, Ricardo de Almeida. Curso de Análise de Sistemas: Notas de Aula. Departamento de Informática, UFES, 2002. Disponível em http://www.inf.ufes.br/~falbo.
GANE, Chris. Análise estruturada de sistemas. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora, 1983.
POMPILHO, S. Análise Essencial – guia prático de Análise de Sistemas. Rio de Janeiro: Editora Ciência Moderna, 2002.
PRESSMAN, Roger S. Engenharia de software. 5ª ed. São Paulo: Makron Books, 2002.
PRESSMAN, Roger S. Engenharia de software. 3ª ed. São Paulo: Makron Books, 1995.
YOURDON, Edward. Análise estruturada moderna. Rio de Janeiro: Campus, 1992.

1. Introdução

Como vimos ao estudarmos o Ciclo de Vida de Software, a Análise de Requisitos ou simplesmente Análise é uma tarefa da Engenharia de Software que efetua a ligação entre a definição do escopo do projeto de software dentro de um sistema mais amplo (Levantamento) e o projeto de Software, propriamente.

Esta atividade permite que o engenheiro de software (normalmente chamado de Analista de Sistemas neste papel) aprimore o escopo do software e construa modelos lógicos (e portanto, independentes de aspectos de implementação) do sistema.

Tem como ponto de partida os resultados da atividade de Levantamento (Especificação de Requisitos) e tem como principal propósito gerar uma especificação apresentando a modelagem do sistema através de ferramentas de modelagem (Modelos de Dados, modelos funcionais e modelos de controle).

1.1. O Analista de Sistemas

Assim como a tarefa de Levantamento e especificação de requisitos, a tarefa de Análise de Requisitos é um processo de descoberta, ou seja, de definição de *o quê* o sistema deve fazer para atender às necessidades dos usuários. E assim como no Levantamento, os principais profissionais envolvidos nesta atividade são o Analista e o cliente / usuário.

Portanto, para ter uma boa probabilidade de sucesso, o analista de sistemas deve possuir uma formação que vai além das disciplinas voltadas para o conhecimento de computadores. O seguinte conjunto de habilidades seria adequado para o bom desempenho da atividade de análise de sistemas:

- *capacidade de comunicação:* entendida como a capacidade de ouvir, redigir e expor idéias com clareza e precisão, aprender e expressar o conteúdo do aprendizado com facilidade;
- *capacidade de análise:* entendida como a aptidão para realizar operações mentais com abstrações sobre o recorte da realidade em estudo. Possuir uma visão sistêmica e crítica do contexto em análise;
- *conhecimento da área usuária:* entendido, principalmente, como aquele tipo de conhecimento que não pode ser obtido através de treinamento, mas adquirido por meio da experiência pessoal ao longo da vivência com as operações da empresa;
- *capacidade de negociação:* entendida como a habilidade em obter o resultado desejado, ainda que em condições adversas, e capacidade de administrar conflitos de interesses interpessoais que surjam durante o trabalho em equipe.
- *administração de projetos:* entendida como noções sobre alocação de tempo e recursos humanos, financeiros e materiais necessários a projetos, procurando sempre a efetividade dos projetos[1].
- *conhecimento técnico:* entendido como a capacidade de se especificar sistemas de informação, principalmente nas suas fases de nível de abstração mais alto, utilizando ferramentas de modelagem.

As habilidades acima colocadas formam um perfil ideal para o Analista. No entanto, é difícil encontrarmos profissionais com todas essas habilidades. Considera-se imprescindível as duas primeiras habilidades, pois sem elas dificilmente se adquirirá as demais.

[1] O conceito de eficiência está normalmente associado a desempenho, ou seja, quanto maior a eficiência, maior o desempenho (em relação a custos, performance ou qualquer outro critério de desempenho). Por outro lado, o conceito de eficácia está normalmente associado ao que produz efeito, ao que dá resultado. Quando se diz que algo tem efetividade, quer se dizer que é eficaz e eficiente. No ciclo de vida de software, a Análise se concentra em modelar um sistema eficaz – que atenda às necessidades do usuário – e o Projeto se concentra em aspectos de eficiência.

O conhecimento dessas habilidades é importante para evitar-se erros cometidos no mercado de trabalho, seja pelas empresas, seja pelos próprios profissionais, a de passar um bom programador ou projetista de sistemas para a atividade de analista de sistemas. Devido aos diferentes perfis exigidos pelas diferentes atividades, pode-se transformar um bom programador ou projetista em um analista medíocre. O mesmo é válido para a situação inversa.

O analista deve ter um perfil adequado a um especificador de sistemas, utilizando as ferramentas de modelagem para isso. Já o projetista e o programador devem ter um perfil adequado a um criador de soluções tecnológicas, utilizando as ferramentas tecnológicas para isso (hardware, software, bancos de dados, linguagens de programação, etc).

Ambos tem importância fundamental em um projeto de desenvolvimento de software e, muitas vezes, um mesmo engenheiro de software (desenvolvedor) possui habilidades para desempenhar de forma satisfatória todos os papéis.

1.2. Princípios de Análise

Durante as últimas décadas, um grande número de métodos de modelagem para a etapa de análise foi desenvolvido. Cada método tem os seus diagramas específicos, suas notações e pontos de vista particulares. No entanto, todos eles estão relacionados por um conjunto de princípios fundamentais:

1. O domínio da informação de um problema precisa ser representado e entendido.

   Todas as aplicações de software podem ser coletivamente chamadas de *processamento de dados*. Esse termo contém um ponto chave para o entendimento que um software é construído para aceitar entradas de dados, processá-los e produzir dados processados como saídas.

   Este primeiro princípio da análise requer um exame do domínio da informação e a criação de um modelo de dados.

2. As funções a serem desenvolvidas pelo software devem ser definidas.

   O software transforma a informação, realizando três funções genéricas: entrada, processamento e saída. Os modelos funcionais, diferentemente dos modelos de dados, focalizam as funções do sistema, normalmente começando com um diagrama de contexto – onde o sistema é representado por uma única função – e sendo refinados até que possamos especificar cada função.

3. O comportamento do software (como conseqüência de eventos externos) precisa ser representado.

   O software responde a eventos do mundo exterior. Esta característica estímulo/resposta forma a base do modelo de controle, que cria uma representação dos estados do software e dos eventos que podem mudar esses estados.

   Os modelos, sejam de dados, funcionais ou de controle, cumprem papéis importantes:

   - ajudam o analista a entender a informação, a função e o comportamento de um sistema, tornando a tarefa de análise de requisitos mais fácil e sistemática;
   - permitem tratar a complexidade do problema por níveis de abstração, começando pela visão mais abstrata e descendo a visões progressivamente mais detalhadas;
   - estabelecem uma visão comum (entre usuário e analista) do sistema, permitindo a discussão de alterações e correções nos requisitos a baixo custo e com mínimo risco;
   - tornam-se o ponto focal para a revisão e, portanto, a chave para a determinação da inteireza, consistência e precisão da especificação do software;
   - tornam-se a base para o projeto, fornecendo ao projetista uma representação essencial do software, que pode ser “mapeada” em um contexto de implementação;

- provêem facilidades para a geração de testes de aceitação;
- formam uma base para a documentação do sistema.

Vale ressaltar ainda que os modelos de dados, funcionais e de controle, terão maior ou menor importância de acordo com o sistema que está sendo especificado, e esses sistemas, por sua vez, podem ser classificados de acordo com essa importância relativa de cada modelo.

- Sistemas centrados em dados, como os sistemas de bancos de dados.
- Sistemas centrados em funções, como os sistemas científicos.
- Sistemas centrados em controle, como os sistemas de tempo-real.

No caso de sistemas de informação administrativos, que são nosso principal objeto de estudo, os aspectos mais importantes são as funções e os dados, principalmente estes últimos.

4. Os modelos que mostram informação, função e controle devem ser particionados de um modo que revele detalhes em forma de camadas.

   Os problemas são freqüentemente muito grandes e complexos para serem entendidos como um todo. Por isso tendemos a particionar (dividir) tais problemas em segmentos que podem ser facilmente entendidos e estabelecer interfaces entre esses segmentos de modo que a função global possa ser entendida.

   Em essência, o particionamento decompõe um problema nos seus segmentos constituintes. Estabelecemos uma representação hierárquica da função ou da informação e depois particionamos o elemento superior (1) expondo cada vez mais detalhes, movendo-nos verticalmente na hierarquia ou (2) decompondo funcionalmente o problema, movendo-nos horizontalmente na hierarquia.

5. O processo de análise deve ir da informação essencial até o detalhe de implementação.

   Uma visão essencial dos requisitos de software apresenta as funções a serem realizadas e a informação a ser processada, sem cuidar de detalhes de implementação. Concentrando-se na essência do problema, durante as tarefas de levantamento, especificação e análise de requisitos, deixamos nossas opções abertas para especificar detalhes de implementação somente durante as tarefas de projeto e codificação.

1.3. Métodos de Análise

Entre os mais importantes métodos de Análise, três se destacam: A Análise Estruturada, a Análise Essencial e a Análise Orientada a Objetos. A seguir discutiremos suas principais características.

- **Análise Estruturada:**

  A Análise Estruturada adota uma visão de desenvolvimento baseada em um modelo entrada-processamento-saída. No paradigma estruturado, os dados são considerados separadamente das funções que os transformam e a decomposição funcional é usada intensamente.

  Funções, a princípio, são ativas e têm comportamento, enquanto dados são repositórios passivos de informação, afetados por funções. Esta divisão tem origem na arquitetura de hardware de von Neumann, onde a separação entre programa e dados é fortemente enfatizada.

  O sistema é decomposto em funções, e os dados são transportados entre elas. Esta é a filosofia da proposta original da Análise Estruturada, cuja ferramenta básica de modelagem são os diagramas de fluxo de dados (DFDs).

  Diagramas de Fluxos de Dados (DFDs): É a ferramenta de modelagem utilizada para descrever a transformação de entradas em um sistema, em saídas. É composto por processos, fluxos de dados, depósito de dados e terminadores (ou entidades externas).

- **Análise Essencial = Análise Estruturada Moderna:**

  Os *métodos orientados a funções* conduzem o desenvolvimento de software estruturando as aplicações segundo a ótica das funções (ações) que o sistema deverá realizar.

  Os *métodos orientados a dados*, por sua vez, enfatizam a identificação e estruturação dos dados, subjugando a análise das funções para um segundo plano. Esses métodos têm origem no projeto de bancos de dados e, geralmente, têm no modelo de Entidades e Relacionamentos (ER) sua principal ferramenta.

  A ênfase nas funções, geralmente, leva a sistemas com muita redundância e, conseqüentemente, inconsistentes e difíceis de serem integrados. Por outro lado, a ênfase nos dados está fundamentada em dois fatores significativos:

  - Dados possuem existência própria nas organizações independentemente dos processos que os manipulam.
  - Dados são muito mais estáveis que as funções em uma organização. A menos que haja grandes mudanças nos negócios de uma empresa, os dados tendem a se manter estáveis.

  Assim, é possível desenvolver modelos de dados sem redundância, sem inconsistência e fáceis de integrar. Entretanto, uma vez que o modelo de dados deve representar a realidade, e o conhecimento da realidade, muitas vezes, passa pelo conhecimento das funções, ele deve ser construído de forma iterativa, não podendo ser considerado um produto acabado.

  A Análise Essencial é também uma metodologia estruturada, sendo uma evolução da Análise Estruturada. Essa metodologia procurou conciliar as abordagens orientadas a dados e a funções em um único método, utilizando modelos para dados, funções e controles (DFDs, DERs e Diagramas de Transição de Estados, respectivamente) como ferramentas para a modelagem de sistemas.

- **Análise Orientada a Objetos:**

  A orientação a objetos pressupõe que o mundo é composto por *objetos*, onde um objeto é uma entidade que combina estrutura de dados e comportamento funcional. No paradigma orientado a objetos, os sistemas são estruturados a partir dos objetos que existem no domínio do problema, isto é, os sistemas são modelados como um número de objetos que interagem. Algumas de suas principais ferramentas de modelagem são:

  Diagrama de Classes/Objetos: Diagrama utilizado para a modelagem de classes de objetos, seus atributos e suas funções.

  Diagrama de Casos de Uso: Utilizado para modelar o comportamento de um sistema ou de parte de um subsistema. Muito útil para o entendimento comum de um sistema por usuários e desenvolvedores. Envolve casos de uso – seqüência de ações que um sistema realiza que resulta em algo observável por um ator e atores – Algo ou alguém fora do sistema que interage com o sistema.

2. Modelagem Funcional

A partir deste momento, passaremos a nos preocupar com a modelagem das funções que o sistema deverá executar para atender aos anseios dos usuários do sistema.

A técnica mais difundida para esta finalidade é a utilização de Diagramas de Fluxo de Dados - DFDs, proposta por Gane e Sarson por De Marco. Muitos outros autores citam esta técnica em suas obras, sendo que destacamos como referência o livro “Análise Estruturada Moderna” de Ed Yourdon.

Além dos Diagramas de Fluxo de Dados, são necessários para uma completa modelagem das funções:

- Dicionário de Dados
- Especificações de Processos ou Mini-especificações – Descrição Lógica dos processos primitivos (que não necessitam ser decompostos).

2.1. DFDs

Ferramenta de modelagem utilizada para descrever a transformação de entradas em saídas. Representa um sistema como uma rede de processos interligados entre si por fluxos de dados e depósitos de dados. É a ferramenta de modelagem mais importante da Análise Estruturada. Junto com o DER, é a ferramenta mais importante da Análise Essencial.

É principalmente importante em sistemas centrados em funções ou que, mesmo centrados em dados ou controle, tenham complexidade funcional significativa.

Seus componentes são: Processos, Fluxos de Dados, Depósitos de Dados e Entidades Externas ou Terminadores.

2.1.1. O Processo

Também chamado de função, bolha ou transformação, o processo representa as transformações realizadas sobre os dados em um sistema e correspondem a funções ou procedimentos que um sistema tem que prover.

É representado graficamente por círculos e deve ter um nome composto por um verbo e um objeto. Ver exemplos da pág. 181 do Yourdon.

2.1.2. O Fluxo de Dados

Representa uma coleção de itens de dados em movimento. É representado graficamente através de uma seta entrando ou saindo de um processo, com a ponta indicando o sentido do fluxo. Deve ter associado um nome o mais significativo possível, de modo a facilitar a validação do diagrama com os usuários. Esse nome deve ser um substantivo que facilite a identificação da coleção de dados transportada.

Ver exemplo da figura 9.5 da pág. 183 do Yourdon.
Observação: Quando falamos em coleção de dados, estamos falando tanto de fragmentos de um pacote de dados – como um item individual de dados (Ex: Nome do Cliente) ou alguns itens individuais de dados combinados (Ex: Nome e CPF do Cliente) – quanto em pacotes de dados completos (Ex: Todos os dados de um Cliente).

Observação: Apesar do Yourdon citar na pág. 182 que DFDs podem ser utilizados para modelar outros sistemas, além dos sistemas de informação, inclusive dando exemplos desse tipo como o da preparação do bolo da pág. 183, em sistemas de informação, não devemos representar fluxos de natureza não informacional, isto é, dinheiro, materiais, etc, e sim, como o próprio nome define, fluxos de **dados**.

Os fluxos de dados podem ser de entrada (em um processo), de saída (de um processo) ou de diálogo (nos dois sentidos). Além de poderem ser divergentes ou ramificados. Ver exemplos das páginas 184, 185 e 186 do Yourdon.

Observação: Um DFD não modela quando e em que seqüência os fluxos de dados se movimentam pelo sistema. Esses detalhes procedurais deverão ser modelados por alguma ferramenta de modelagem procedural, tais como as que veremos para a especificação de processos.

2.1.3. O Depósito de Dados

O depósito de dados é utilizado para modelar um repositório de uma coleção de pacotes de dados em repouso. É representado graficamente por duas linhas paralelas com um nome que represente seu conteúdo e que normalmente é o plural do nome dos pacotes transportados pelos fluxos para dentro e para fora do depósito. Ver exemplos da pág. 188 do Yourdon.

Observação: Não confundir depósito de dados com bancos de dados. Como utilizamos o DFD na etapa de Análise, ou seja, sem preocupações com aspectos de implementação, os depósitos de dados podem representar qualquer repositório identificado em um sistema, eletrônico ou não.

O depósito de dados em um DFD existe devido à natureza assíncrona dos processos, ou seja, quando um processo gera um fluxo de dados de saída que será um fluxo de dados de entrada para um outro processo, este último pode não

processá-lo imediatamente, o que exige o armazenamento dos dados transportados pelo fluxo para futuro processamento. Ver exemplos das páginas 190 e 191 do Yourdon.

Um fluxo de dados que sai de um depósito é chamado de leitura, ou seja, o conteúdo do depósito não se alterará em função dele.

Um fluxo de dados que entra em um depósito é chamado de escrita, ou seja, o conteúdo do depósito se alterará em função dele. Um fluxo de escrita pode ter 3 significados:

- uma inclusão, quando um ou mais novos pacotes estão sendo introduzidos no depósito.
- uma exclusão, quando um ou mais pacotes estão sendo eliminados ou removidos do depósito.
- uma alteração, quando um ou mais pacotes estão sendo modificados ou alterados. Isso pode envolver a alteração de todo um pacote ou apenas de fragmentos dele (que é o caso mais comum).

Quando examinamos fluxos de dados que entram ou saem de um depósito, surge uma dúvida: o fluxo representa um único pacote, múltiplos pacotes, fragmentos de um pacote, ou fragmentos de vários pacotes de dados? Em algumas situações, essas dúvidas podem ser respondidas pelo simples exame do rótulo do fluxo e, para tal, adotamos a seguinte convenção:

- se um fluxo não possuir rótulo ou tiver o mesmo rótulo do depósito de dados, então um pacote inteiro de informação ou múltiplas instâncias do pacote inteiro estão trafegando pelo fluxo;
- se o rótulo de um fluxo nomeado for diferente do rótulo do depósito, então as informações que estão trafegando são um ou mais componentes (fragmentos) de um ou mais pacotes, e estarão definidas no dicionário de dados.

Observação: Em DFDs complexos, para se evitar cruzamentos de setas de fluxos de dados, pode-se desenhar um mesmo depósito de dados mais de uma vez.

Ver exemplos das pág. 191, 192, 193 e 194 do Yourdon.

2.1.4. Terminador ou Entidade Externa

Entidades externas ou terminadores são fontes ou destinos de dados do sistema. Representam os elementos do ambiente com os quais o sistema se comunica. Tipicamente, uma entidade externa é uma pessoa (p.ex. um cliente), um grupo de pessoas (p. ex. um departamento da empresa ou outras instituições) ou um outro sistema que interaja com o sistema em questão. Uma entidade externa deve ser identificada por um nome e representada por um retângulo. Assim como no caso dos depósitos de dados, em diagramas complexos, podemos desenhar um mesmo terminador mais de uma vez, para se evitar o cruzamento de linhas de fluxos de dados ou para impedir que longas linhas de fluxos de dados saiam de um lado a outro do diagrama. Ver exemplo da pág. 194 do Yourdon.

Ao identificarmos alguma coisa ou sistema como uma entidade externa, estamos afirmando que esta entidade está fora dos limites do sistema em questão e, portanto, fora do controle do sistema que está sendo modelado. Assim, qualquer relacionamento existente entre entidades externas não deve ser mostrado em um DFD.

Se percebermos que, em algum ponto do sistema, descrevemos algo que ocorre dentro de uma entidade externa ou relacionamentos entre entidades externas, é necessário reconhecer que a fronteira do sistema é na realidade mais ampla do que foi estabelecido inicialmente e, portanto, deve ser revista.

Uma vez que os terminadores são externos ao sistema, os fluxos de dados que os interligam aos diversos processos representam a interface entre o sistema e o mundo externo.

2.1.5. Diretrizes para a Elaboração de um DFD

Para construir DFDs corretos e agradáveis e, portanto, mais fáceis de serem entendidos e validados por outros participantes do projeto, devemos seguir algumas diretrizes:

- Escolher nomes significativos para os processos, fluxos, depósitos e terminadores.
    - Combinação verbo + objeto para os processos.

  - Evitar nomes vagos.
  - Usar nomes significativos para o usuário.
- Numerar os processos de maneira consistente.
- Refazer os DFDs quantas vezes forem necessárias.
- Evitar DFDs complexos demais, expandindo-o em níveis, quando necessário.
  - DFDs devem evitar grandes dimensões ou complexidade (com 7 ± 2 processos e que caibam em uma folha de papel comum). Exceções: Diagrama de Contexto (pode ter muitos terminadores) e DFD 0 (pode ter muitos processos).
  - Os DFDs devem ser expandidos até que os processos possam ser claramente especificados (± 1 página de especificação).
- Certificar-se que o DFD é logicamente consistente.
  - Evitar "poços sem fundo" (processos com entradas, mas sem saídas).
  - Evitar processos com geração espontânea (processos com saídas, mas sem entradas).
  - Tomar cuidado com depósitos de leitura apenas ou de escrita apenas.

Ler e verificar atentamente o texto e os exemplos das páginas 196 a 205 do Yourdon.

2.1.6. DFD com Níveis

Como já mencionado no estudo sobre processos, é uma boa prática manter um certo nível de complexidade nos processos representados em um DFD. Esse nível de complexidade pode ser estabelecido pelo tamanho da especificação da lógica do processo ou pelo número de processos em um diagrama. Se tal nível de complexidade for superado, devemos organizar o DFD em níveis:

Dois DFDs especiais estão presentes em qualquer modelo funcional:

- **Contexto**: mostra o sistema como uma "caixa-preta", trocando informações (fluxos de dados) com entidades externas ao sistema. Define o escopo de abrangência do sistema, indicando que se está renunciando à possibilidade de examinar qualquer coisa além da fronteira definida pelas entidades externas. Ver exemplo 9.19 da pág. 206 do Yourdon.
- **0 (Zero) – Geral ou de Sistema**: é uma decomposição do diagrama de contexto, mostrando o funcionamento do sistema em questão, isto é, as grandes funções do sistema e as interfaces entre elas. Os processos nesse diagrama recebem os números 1, 2, 3, etc... É necessário assegurar a coerência entre os diagramas **C** e **0**, isto é, assegurar que os fluxos de dados entrando ou saindo do diagrama **0** efetivamente reproduzem as entradas e saídas do diagrama **C**. Neste diagrama, devem aparecer os depósitos de dados necessários para a sincronização dos processos.

Os processos do DFD 0 que sejam complexos, devem ser expandidos, dando origem a DFDs detalhados. Os processos dos DFDs detalhados que ainda possuírem complexidade que não permita sua especificação de forma procedural, também deverão ser expandidos em um nível maior de detalhe, e assim sucessivamente, até que se possa especificar todos os processos. Em cada expansão, deve-se assegurar a coerência entre o diagrama detalhado e o diagrama de nível superior.

Nesse ponto que a numeração dos processos se torna essencial, pois o diagrama detalhado de um processo deverá ser nomeado com o número e o nome do processo que está sendo detalhado e seus processos deverão ser numerados com o número do processo que está sendo detalhado (p. ex., 2) e um número seqüencial, separados por um ponto (p. ex., 2.1, 2.2, etc.). Ver exemplo da pág. 208 do Yourdon.

Ler e verificar atentamente o texto e os exemplos das páginas 205 a 214 do Yourdon.

2.2. Dicionário de Dados

O Dicionário de Dados é uma listagem organizada de todos os elementos de dados pertinentes ao sistema, com definições precisas para que os usuários e desenvolvedores possam conhecer o significado de todos os itens de dados manipulados pelo sistema. Descreve o significado, a composição, os valores e as unidades significativas de fluxos de dados, depósitos de dados, entidades e relacionamentos através de estruturas de dados e de elementos de dados.

- Estrutura de Dados: Representam fluxos, depósitos, entidades ou relacionamentos. São compostas de elementos de dados e/ou de outras estruturas.
- Elemento de Dados: São os elementos das estruturas que não necessitam mais de decomposição.

Notação:

Existem diferentes notações em uso. Neste curso, adotaremos a de Tom DeMarco.

Símbolo	**Significado**
=	é composto de
+	e
( )	estruturas ou elementos de dados opcionais
[|]	estruturas ou elementos de dados alternativos
n{ }m	repetição de estruturas ou elementos de dados, onde *n* representa o número mínimo de repetições e *m* o número máximo. Se *n* e *m* não forem especificados, significa que 0 ou mais repetições.
**	delimitadores de comentários
_____ ou @	identificador/atributo determinante

Exemplificando para uma estrutura de dados “nome”:

nome =	título-cortesia + primeiro-nome + (nome-intermediário) + último-nome
título-cortesia =	[Sr. | Sra. | Srta. | Dr. | Professor]
primeiro-nome =	1{caracter válido}
nome-intermediário =	1{caracter válido}
último-nome =	1{caracter válido}
caracter-válido =	[A-Z | a-z | 0-9 | ‘ |]

No exemplo, “nome” é uma estrutura de dados e caracter-válido é um elemento de dados.

Os exemplos mostrados a seguir ilustram diversas situações para a estrutura de dados CLIENTES (neste caso, um depósito de dados de um DFD que corresponderá a uma entidade CLIENTE do DER) e o emprego das notações.

(a) O cliente pode possuir um telefone.

CLIENTES = **clientes da livraria**
código-cliente + nome-cliente + endereço-cliente + (telefone-cliente)

(b) O cliente pode possuir zero ou mais telefones.

CLIENTES = **clientes da livraria**
código-cliente + nome-cliente + endereço-cliente + {telefone-cliente}

(c) O cliente pode possuir até três telefones.

CLIENTES = **clientes da livraria**
código-cliente + nome-cliente + endereço-cliente + {telefone-cliente}3

(d) O cliente pode possuir telefone comercial, residencial ou ambos.

CLIENTES = **clientes da livraria**
código-cliente + nome-cliente + endereço-cliente + [telefone-comercial | telefone-residencial | telefone-comercial + telefone-residencial]

2.3. Especificação de Processos

Quando chegamos a um nível de especificação em que os processos não são mais decomponíveis, precisamos complementar esta especificação com descrições das lógicas desses processos. A especificação de processos deve ser feita de forma que possa ser validada por analistas e usuários. Entretanto, encontramos muitos problemas na descrição de forma narrativa, entre os quais podemos citar:

- Uso de expressões do tipo: mas, todavia, a menos que ...

  Por exemplo, qual a diferença entre as declarações abaixo ?
  - Somar A e B, a menos que A seja menor que B, onde, neste caso, subtrair A de B.
  - Somar A e B. Entretanto, se A for menor que B, a resposta será a diferença entre B e A.
  - Somar A e B, mas subtrair A de B quando A for menor que B.
  - total é a soma de B e A. Somente quando A for menor que B é que a diferença deve ser usada como o total.

Ao analisarmos estas frases, notamos que não existe diferença lógica entre elas, entretanto as formas narrativas apresentadas mascaram a semelhança existente. Se ao invés de usarmos uma forma narrativa, usarmos uma forma padrão do tipo *se-então-senão*, teremos maior clareza e validação.

```
se A < B
        então TOTAL = B - A;
        senão TOTAL = A + B;
fim-se;
```

- Uso de comparativos como: Maior que / Menor que, Mais de / Menos de.

  Seja a seguinte declaração: “Até 20 unidades, sem desconto. Mais de 20, 5% de desconto.”

  E exatamente 20 unidades, que tratamento deve ser dado ?

- Ambigüidades do E/OU.

  Seja a seguinte declaração: “Clientes que gerarem mais de um milhão de cruzeiros em negócios por ano **e** possuírem um bom histórico de pagamentos **ou** que estiverem conosco há mais de 20 anos, devem receber tratamento prioritário.”

  Quem deverá receber tratamento prioritário? Clientes com mais de 1 milhão em negócios por ano que possuírem bom histórico de pagamentos? Clientes com mais de 20 anos? Clientes com mais de 1 milhão e (ou bom histórico, ou mais de 20 anos)?

  Note que pela declaração não fica claro quando deverá ser aplicado o tratamento prioritário.

- Uso de Adjetivos Indefinidos

  Na declaração do item anterior, o que é um ***bom*** histórico de pagamentos? Devemos tomar cuidado ao utilizarmos adjetivos indefinidos e quando o fizermos, tomarmos o cuidado de defini-los.

Para administrar os problemas oriundos da narrativa, são utilizadas técnicas de especificação de processos, entre as quais podemos citar:

- Português Estruturado
- Tabelas de Decisão
- Árvores de Decisão
- Combinação das técnicas acima

2.3.1. Português Estruturado

O Português Estruturado é um subconjunto do Português, cujas sentenças são organizadas segundo as três estruturas de controle introduzidas pela Programação Estruturada: seqüência, seleção e repetição.

- **Instruções de Seqüência:** grupo de instruções a serem executadas que não tenham repetição e não sejam oriundas de processos de decisão. São escritas na forma imperativa, como no exemplo abaixo.

  obter ...
  atribuir ...
  armazenar ...

- **Instruções de Seleção**: quando uma decisão deve ser tomada para que uma ação seja executada, utilizamos uma instrução de seleção. As instruções de seleção são expressas como uma combinação *se-então-senão*, conforme abaixo.

  se <condição>
  então grupo_ de_ações_1;
  senão grupo_de_ações_2;
  fim-se;

  Exemplo:

  se Número_de_Dependentes = 0
  então Salário_Família = 0;
  senão Salário_Família = Salário_Mínimo / 3;
  fim-se;

  Quando existirem várias ações dependentes de uma mesma condição, que sejam mutuamente exclusivas, podemos utilizar uma estrutura do tipo *caso*, conforme abaixo.

  caso <condição> =
  valor_1 : grupo_de_ações_1;
  valor_2 : grupo_de_ações_2;
  . . . . . .
  valor_n : grupo_de_ações-N;
  fim-caso;

  Exemplo:

  caso opção =
  1 : incluir novo cliente;
  2 : excluir cliente existente;
  3 : alterar dados de cliente;
  senão : executar rotina de erro;
  fim-caso;

- **Instruções de Repetição**: Aplicadas quando devemos executar uma instrução, ou um grupo de instruções, repetidas vezes. A estrutura de repetição pode ser usada de três formas distintas:

  1. **para cada "X" faça**
     grupo_de_ações;
     fim-para;

     Exemplo:

     para cada Aluno faça

```
        Média = (Prova_1 + Prova_2) / 2;
        imprima Média;
fim-para;
```

2. **enquanto <condição for verdadeira> faça**
   grupo_de_ações;
   fim-enquanto;

   Exemplo:

```
enquanto existir registro faça
        ler registro;
        consistir dados;
fim-enquanto;
```

3. **repita**
   grupo_de_ações;
   até que <condição seja verdadeira>;

   Exemplo:

```
repita
        ler registro
        consistir dados
até que todos os registros do arquivo tenham sido processados;
```

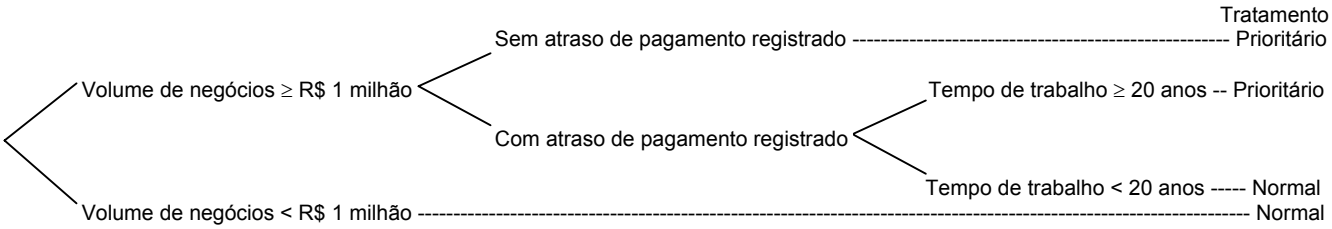
Uma especificação de processo em Português Estruturado deve possuir as seguintes características gerais:

- deve ser clara, concisa, completa e livre de ambigüidades;
- todos os dados citados na especificação que estejam definidos no dicionário de dados devem ser sublinhados;
- os dados definidos localmente não são sublinhados;
- os depósitos de dados, além de sublinhados, devem ser escritos com letras maiúsculas;
- sempre que um comando de seleção ou repetição for utilizado, os comandos do bloco interno (grupo_de_ações) devem estar identados, de modo a dar a clareza de que esses comandos fazem parte das ações da seleção ou repetição.

2.3.2. Árvore de Decisão

Árvores de Decisão são excelentes para mostrar a estrutura de decisão de um processo. Os ramos da árvore correspondem a cada uma das possibilidades lógicas. É uma excelente ferramenta para esquematizar a estrutura lógica e para obter do usuário a confirmação de que a lógica expressada está correta. De forma clara e objetiva, permite a leitura da combinação das circunstâncias que levam a cada ação.

Como podemos notar pelo exemplo abaixo, uma Árvore de Decisão é muito boa para representar a lógica decisória. Entretanto, se for necessário descrever a lógica de um processo como um conjunto de instruções, combinando decisões e ações intermediárias, a árvore de decisão deve ser preterida em favor do português estruturado ou combinada a ele.



2.3.3. Tabelas de Decisão

Tabelas de decisão são usadas em aplicações semelhantes às das árvores de decisão. As árvores de decisão são mais indicadas, quando o número de decisões for pequeno e nem todas as combinações de condições forem possíveis. As tabelas de decisão aplicam-se melhor a situações em que o número de ações é grande e ocorrem muitas combinações de condições. Também devemos utilizar tabelas de decisão se existirem dúvidas de que a árvore de decisão não mostra toda a complexidade do problema.

O formato básico de uma tabela de decisão é mostrado na figura abaixo.

Nome da Tabela	
Condições	Combinações
Ações	Regras

A construção de uma tabela de decisão envolve os seguintes passos:

1. Levantar as ações do processo;
2. Identificar as condições que determinam estas ações;
3. Identificar os estados possíveis de cada condição;
4. Identificar as combinações dos estados das condições;
5. Construir uma coluna para cada combinação de condições;
6. Preencher cada coluna com as regras das ações correspondentes;
7. Verificar se o entendimento foi correto;
8. Alterar a tabela até obter total concordância dos usuários;
9. Se possível, compactar a tabela.

Em função do tipo das condições, temos dois tipos de tabelas:

- Tabela de Entrada Limitada: os valores de uma condição se limitam a dois. Exemplos típicos deste tipo de tabelas são as tabelas cujas condições são escritas sob a forma de perguntas, de modo que as respostas sejam “sim” ou “não”, como mostra o exemplo abaixo.

Tratamento de Clientes								
Volume de Negócios ≥ R$ 1 milhão?	S	S	S	S	N	N	N	N
Atraso de pagamento registrado?	N	N	S	S	N	N	S	S
Tempo de trabalho ≥ 20 anos?	S	N	S	N	S	N	S	N
Tratamento prioritário	X	X	X					
Tratamento normal				X	X	X	X	X

- Tabela de Entrada Ampliada: Uma condição pode ter mais de dois estados diferentes, como no exemplo abaixo.

Cobrança de Fretes												
Meio de Transporte	F	F	F	F	R	R	R	R	M	M	M	M
Tipo de Entrega	R	R	N	N	R	R	N	N	R	R	N	N
Peso	L	P	L	P	L	P	L	P	L	P	L	P
R$ 100/Kg					X				X			
R$ 50/Kg	X					X	X			X	X	
R$ 10/Kg		X	X	X				X				X

Meio de Transporte: Ferroviário (F), Rodoviário (R), Marítimo (M).
Tipo de Entrega: Rápida (R) – até 5 dias úteis; Normal (N) – até 30 dias.
Peso: Leve (L): ¡Ü 100kg; Pesado (P): > 100Kg

Muitas vezes, grupos de condições levam à mesma ação. Para estes casos, podemos utilizar tabelas compactadas, como a do exemplo abaixo.

Tratamento de Clientes				
Volume de Negócios ≥ R$ 1 milhão?	S	S	S	N
Atraso de pagamento registrado?	N	S	S	-
Tempo de trabalho ≥ 20 anos?	-	S	N	-
Tratamento prioritário	X	X		
Tratamento normal			X	X

3. Análise Essencial

A etapa de análise de requisitos, geralmente chamada de análise de sistemas, é um processo de comunicação entre engenheiros de software (analistas de sistemas) e clientes/usuários do sistema, com o objetivo de definir detalhadamente o propósito e os requisitos de um software. Os requisitos de um sistema compreendem o conjunto de características que o sistema deve possuir para atingir seu propósito.

A análise de sistemas é um processo de transmissão de conhecimento e, assim sendo, envolve três etapas:

- aprendizado: aprender sobre o domínio do problema onde o sistema será inserido;
- estruturação e representação dos requisitos do sistema: consiste na modelagem do sistema propriamente dita;
- validação dos requisitos com o usuário.

Ao longo do processo, o analista enfrenta o desafio de "lidar com a complexidade", isto é, situações complexas do mundo real devem ser entendidas e representadas de forma simples, para facilitar a compreensão e validação. Para tal, é preciso delimitar a área de estudo, subdividir o todo complexo em partes inteligíveis e gerenciáveis, extrair as características essenciais da realidade e modelar essas características para mostrar o relacionamento entre seus componentes.

A análise de sistemas é, em última instância, uma atividade de construção de modelos. Um modelo é uma representação de alguma coisa do mundo real, uma abstração da realidade, ou seja, representa uma seleção de características do mundo real, que são relevantes para o propósito com o qual o modelo foi construído.

Modelos são ferramentas fundamentais no desenvolvimento de sistemas. Sistemas são modelados para:

- possibilitar o estudo do comportamento do sistema;
- facilitar a comunicação entre os componentes da equipe de desenvolvimento (e clientes e usuários);
- possibilitar a discussão de correções e modificações com o usuário;
- formar uma documentação do sistema.

Um modelo enfatiza um conjunto de características da realidade, que corresponde à *dimensão do modelo*. Além da dimensão que um modelo enfatiza, modelos possuem níveis de abstração. O *nível de abstração* de um modelo diz respeito ao grau de detalhamento com que as características do sistema são representadas. Em cada nível há uma ênfase seletiva nos detalhes representados. No caso dos sistemas de informação, geralmente, são considerados três níveis:

- *conceitual*: considera características do sistema independentes do ambiente computacional (hardware e software) no qual o sistema será implementado. Essas características são dependentes unicamente das necessidades do usuário.
- *lógico*: características dependentes de um determinado *tipo* de sistema computacional. Essas características são, contudo, independentes de produtos específicos.
- *físico*: características dependentes de um sistema computacional específico, isto é, uma linguagem e um compilador específico, um sistema gerenciador de bancos de dados específico, o hardware de um determinado fabricante, etc.

Nas primeiras etapas do processo de desenvolvimento (levantamento de requisitos e análise), o engenheiro de software

representa o sistema através de modelos conceituais. Nas etapas posteriores, as características lógicas e físicas são representadas em novos modelos.

O método de Análise Essencial de Sistemas preconiza que, de uma forma geral, um sistema deve ser modelado através de três dimensões:

- *dados*: diz respeito aos aspectos estáticos e estruturais do sistema;
- *controle*: leva em conta aspectos temporais e comportamentais do sistema;
- *funções*: considera a transformação de valores.

Em relação ao grau de abstração, a Análise Essencial considera dois níveis: o nível essencial e o nível de implementação, representados, respectivamente, pelos seguintes modelos:

- *Modelo Essencial*: representa o sistema num grau de abstração completamente independente de restrições tecnológicas.
- *Modelo de Implementação*: passa a considerar as restrições tecnológicas impostas pela plataforma de hardware e software a ser utilizada para implementar o sistema.

Podemos perceber que o modelo de implementação não corresponde a um modelo de análise propriamente dito, uma vez que considera aspectos de implementação, característica marcante da fase de projeto. De fato, na abordagem da Análise Essencial, este modelo corresponde a uma espécie de zona nebulosa entre as fases de análise e de projeto. Por considerarmos que um modelo considerando aspectos da plataforma de implementação é melhor caracterizado na fase de projeto, neste texto, não trataremos do modelo de implementação.

Durante muito tempo, houve grandes debates entre os profissionais de desenvolvimento de sistemas sobre por qual perspectiva se deveria começar a especificação de um sistema: pelos dados ou pelas funções? Os argumentos, igualmente válidos, exploravam considerações como:

- dados são mais estáveis que funções...,
- sem um entendimento das funções a serem desempenhadas pelo sistema, como definir o escopo e os dados necessários?

A Análise Essencial procurou estabelecer um novo ponto de partida para a especificação de um sistema: a identificação dos *eventos* que o afetam.

Um dos problemas mais relevantes na especificação é como efetuar seu particionamento. A Análise Estruturada propõe um particionamento através de uma abordagem *top-down*. Embora esta seja uma boa maneira de se atacar um problema complexo – começando da visão geral e ir descendo, passo a passo, numa visão hierárquica, a níveis de detalhes cada vez maiores – na prática, esta abordagem não se mostrou eficiente como estratégia de projeto para a decomposição de sistemas. A Análise Essencial propõe uma outra forma de particionamento, a qual é baseada nos eventos, e que tem demonstrado ser mais efetiva do que a abordagem *top-down*, pois torna mais fácil a identificação das funções e entidades que compõem o sistema.

A Análise Essencial de Sistemas, através da técnica de particionamento por eventos, oferece uma boa estratégia para modelar o comportamento do sistema, visando satisfazer os requisitos do usuário, pressupondo-se que dispomos de tecnologia perfeita[2] e que ela pode ser obtida a custo zero.

Apesar de introduzir novos conceitos e novas abordagens, a Análise Essencial preservou todos os modelos da Análise Estruturada. De fato, embora diferentes, a melhor maneira de encarar a Análise Essencial é considerá-la uma evolução da Análise Estruturada. A seguir, os principais conceitos da Análise Essencial são apresentados.

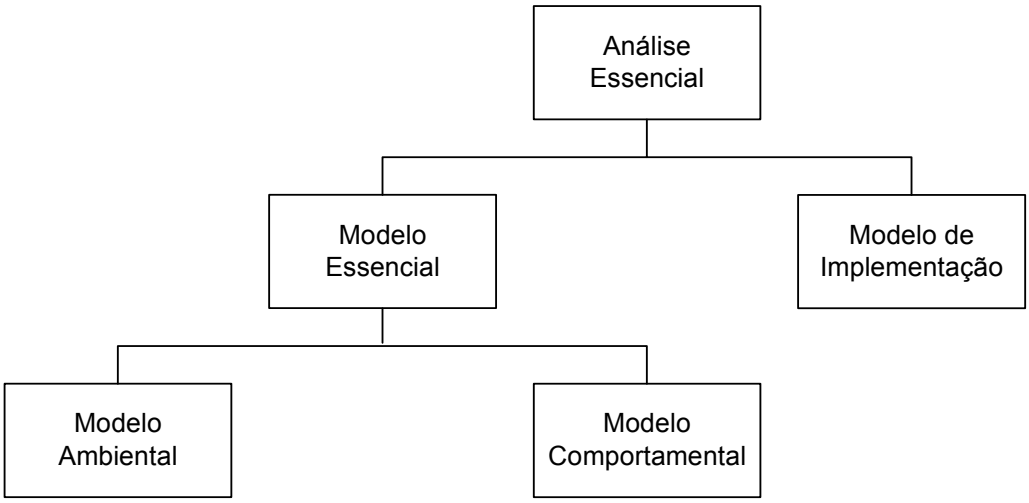
[2] A tecnologia perfeita não possui limitações, isto é, existe um processador perfeito, capaz de executar qualquer processamento, tudo instantaneamente, sem qualquer custo, sem consumir energia, sem gerar calor, sem jamais cometer erros ou parar de funcionar, e um repositório perfeito, capaz de armazenar quantidades infinitas de dados e de ser acessado instantaneamente por qualquer processador, da forma que for mais conveniente.

3.1. Especificação da Essência do Sistema

A Análise Essencial sugere a construção de dois modelos principais, o modelo essencial e o modelo de implementação. Conforme discutido anteriormente, entendemos que apenas o modelo essencial deve ser objeto da fase de análise e, assim, discutiremos apenas a especificação da essência do sistema.

A especificação da essência do sistema, produto da fase de análise, é composta de dois modelos, como mostra a figura abaixo:

- Modelo Ambiental: define a fronteira entre o sistema e o resto do mundo.
- Modelo Comportamental: define o comportamento das partes internas do sistema necessário para interagir com o ambiente.



A Análise Essencial e seus Modelos.

3.2. O Modelo Ambiental

Define a fronteira entre o sistema e o resto do mundo, isto é, o ambiente. Modela a parte *exterior* do sistema, o modelo do *interior* do sistema é o modelo *comportamental*, que será visto posteriormente.

Além de definir o que está dentro e o que está fora do sistema, também define as *interfaces* entre o sistema e o ambiente, ou seja, quais são as entidades presentes no ambiente, porém externas ao sistema que interagem com ele, quais são as informações que entram no sistema provenientes do ambiente externo e quais são as informações produzidas pelo sistema como saídas para o ambiente externo.

Naturalmente, entradas e saídas não são produzidas ao acaso, sendo respostas a eventos ocorridos no ambiente. Desse modo, um outro aspecto básico do modelo ambiental é a identificação dos eventos que ocorrem no ambiente aos quais o sistema deve reagir. Não estamos interessados em todos os eventos que ocorrem no ambiente, estamos interessados apenas nos eventos que exigem uma resposta do sistema.

Observe que a determinação da fronteira entre um sistema e seu ambiente é *arbitrária*, sendo normalmente estabelecida pelo cliente, porém, sendo algo que o analista, habitualmente, poderá influenciar. Veja exemplos das páginas 410 a 412 do Yourdon.

O Modelo Ambiental possui os seguintes componentes:

- Declaração de Objetivos
- Diagrama de Contexto
- Lista de Eventos

3.2.1. Declaração de Objetivos

Enuncia a finalidade do sistema. Pode ser acompanhado de uma breve descrição do contexto do sistema (modelo descritivo ou mini-mundo).

A declaração de objetivos do sistema deve ser elaborada em poucas frases, simples e precisas, em linguagem destituída de vocabulário técnico, de modo a ser entendida pelos usuários do sistema e pela administração da empresa, em geral. Não deve fornecer detalhes sobre como o sistema deverá operar. Veja exemplo das páginas 414 e 415 do Yourdon.

3.2.2. Diagrama de Contexto

É um caso especial do diagrama de fluxo de dados que representa o sistema como um único processo e suas interações com o ambiente. Pode ser acompanhado de um dicionário de dados.

O diagrama de contexto realça diversas características importantes do sistema:

- As pessoas, organizações ou sistemas com os quais nosso sistema comunica-se – conhecidos como terminadores ou entidades externas.
- Os dados que nosso sistema recebe do mundo exterior e que devem ser processados de alguma maneira.
- Os dados produzidos pelo nosso sistema e enviados para o mundo exterior.
- Os depósitos de dados compartilhados entre o nosso sistema e os terminadores.
- Os limites entre o nosso sistema e o resto do mundo.

Veja exemplo da página 416 do Yourdon.

Considerações sobre diagramas de contexto:

- O nome do processo deve ser o nome dado ao sistema. Veja exemplos da página 422 do Yourdon.
- Deve-se representar as entidades externas pelo papel que elas desempenham em relação ao sistema, e não pela pessoa ou entidade específica que o faz. Veja exemplo da página 425 do Yourdon.
- Não pode haver fluxos de dados entre as entidades externas. O que ocorre externamente ao sistema não deve ser representado ou deve-se questionar a fronteira do sistema. Veja exemplo da página 423 do Yourdon.
- Não se deve confundir as entidades externas com o mecanismo utilizado para enviar um fluxo de dados (manipulador). Veja exemplo da página 426 do Yourdon.

3.2.3. Lista de Eventos

A elaboração da lista de eventos é o passo principal desta etapa da Análise Essencial, uma vez que os eventos constituem a parte fundamental de um sistema nessa metodologia.

Evento e resposta são nomes genéricos de interações entre o ambiente externo e o sistema. Um *evento* pode ser definido informalmente como um acontecimento do mundo exterior que requer do sistema uma resposta. Corresponde a alguma mudança no ambiente externo que funcionará como um *estímulo* para o sistema, isto é, o sistema deve responder a este estímulo para atender aos seus objetivos. Uma *resposta* é o resultado da execução de um conjunto de ações no sistema, como conseqüência do reconhecimento pelo sistema de que um evento ocorreu. Uma resposta tipicamente pode ser:

- um fluxo de dados saindo do sistema para uma entidade externa;
- uma mudança de estado em um depósito de dados (inclusão, exclusão ou alteração);
- um fluxo de controle saindo de uma função para ativar uma outra.

Quando um evento ocorre, é produzido um estímulo para o sistema. Ao receber o estímulo, o sistema compreende que o evento ocorreu e ativa os processos necessários para produzir a resposta. Veja exemplo da página 417 do Yourdon.

Os eventos são classificados em três tipos diferentes, dependendo da maneira como ocorrem e da natureza do estímulo que produzem:

- **Evento orientado por fluxo de dados**: é provocado por uma entidade externa, a qual envia dados para o sistema. O estímulo produzido por esse tipo de evento é a chegada de um fluxo de dados que vai ativar uma função. Os eventos externos são nomeados da seguinte forma:

  (*Entidade externa que provocou o evento*) +
  (*ação* – verbo na voz ativa) +
  (*estímulo* – fluxo de dados enviado ao sistema)

  Ex.: Cliente envia pedido.
  Cliente cancela pedido.

- **Evento temporal**: é aquele em que o estímulo é a chegada ao sistema da informação de haver passado um determinado intervalo de tempo. Esses eventos estimulam as ações que o sistema tem de executar em datas previamente conhecidas, isto é, diariamente, mensalmente, etc (o tempo passa e chega o momento do sistema fazer alguma coisa). Pode haver fluxos de dados complementares associados ao evento, mas não é através da chegada destes que o sistema toma conhecimento da ocorrência do evento. Os eventos temporais podem ser nomeados como abaixo:

  (*É hora de*) + (*ação*) + (*complemento*)

  Ex.: Mensalmente, emitir relatório de vendas para a Direção. ou
  É hora de emitir relatório mensal de vendas para a Direção.

- **Evento orientado por controle**: o estímulo provocado por este evento é a chegada ao sistema de um fluxo de controle, o qual apenas notifica o sistema da ocorrência do evento. Pode haver fluxos de dados complementares associados ao evento, mas não é através da chegada do fluxo de dados que o sistema toma conhecimento da ocorrência do evento. Esse tipo de evento pode ser nomeado tendo por base a origem do evento:

  (*Entidade externa que provocou o evento*) +
  (*ação* – verbo na voz ativa) +
  (*complemento*)

  Ex.: Gerente solicita relação de clientes.
  Diretoria autoriza o pagamento de uma fatura.

É importante observar que nem todos os fluxos de dados de um diagrama de contexto estão associados a um evento. Por exemplo, quando um evento ocorre associado a um fluxo de dados proveniente de um terminador, outros fluxos de dados podem ser exigidos pelo sistema de outros terminadores para produzir alguma resposta. Veja exemplo das páginas 417 e 418 do Yourdon.

O que devemos fazer primeiro, a lista de eventos ou o diagrama de contexto? Pode-se iniciar com a lista de eventos ou com o diagrama de contexto, isto não importa, o que importa é que eles estejam consistentes entre si.

A partir do trabalho de levantamento de requisitos realizado, podemos identificar a quais eventos do mundo exterior nosso sistema deverá responder e, uma vez definidos os eventos, é possível construir o Diagrama de Contexto do sistema, mostrando como ele responde a todos os eventos externos relevantes.

Alternativamente, pode-se identificar todas as entidades externas ao sistema, todos os fluxos de entrada e saída do sistema e, conseqüentemente, desenhar o diagrama de contexto, sem termos ainda a lista de eventos. De qualquer maneira, é necessário que se identifique a lista de eventos antes da construção do modelo comportamental.

3.3. O Modelo Comportamental

O modelo comportamental determina o comportamento interno do sistema para que este possa interagir corretamente com o ambiente. Após a construção do modelo comportamental, teremos os seguintes artefatos:

- **Diagrama de Entidades e Relacionamentos**
- **Diagramas de Fluxos de Dados Preliminar (DFD Particionado por Eventos)**: Para cada evento do sistema, deve ser construído um DFD.
- **Diagramas de Fluxos de Dados Organizados em Níveis Hierárquicos**: representa os processos em níveis hierárquicos, a partir do diagrama zero.
- **Diagramas de Transição de Estados**: Representa o comportamento das entidades e relacionamentos com atributos ao longo do tempo. Será construído um DTE para cada entidade ou relacionamento com atributo do DER que possuir comportamento significativo, isto é, possuir mais de um estado ao longo de seu ciclo de vida.
- **Dicionário de Dados**: descreve os dados representados no DER, nos DFDs e nos DTEs.
- **Especificação da Lógica dos Processos (Mini-especificações)**: descreve a lógica dos processos do DFD que não foram detalhados em diagramas de nível inferior (lógica dos processos primitivos).

Como podemos perceber, a Análise Essencial faz uso praticamente das mesmas técnicas de modelagem da Análise Estruturada, a saber a Modelagem de Dados (utilizando modelos de Entidades e Relacionamentos), a Modelagem Funcional (utilizando Diagramas de Fluxo de Dados – DFDs) e a Modelagem de Controle (utilizando Diagramas de Transição de Estados). Isso é bastante natural, já que a Análise Essencial é, de fato, uma extensão da Análise Estruturada.

Na realidade, a principal diferença entre a Análise Essencial e a Análise Estruturada está na estratégia para atacar o problema: a primeira defende uma abordagem baseada em eventos, onde a Análise de Eventos passa a ser um passo fundamental, a segunda é baseada apenas na decomposição *top-down* da funcionalidade do sistema.

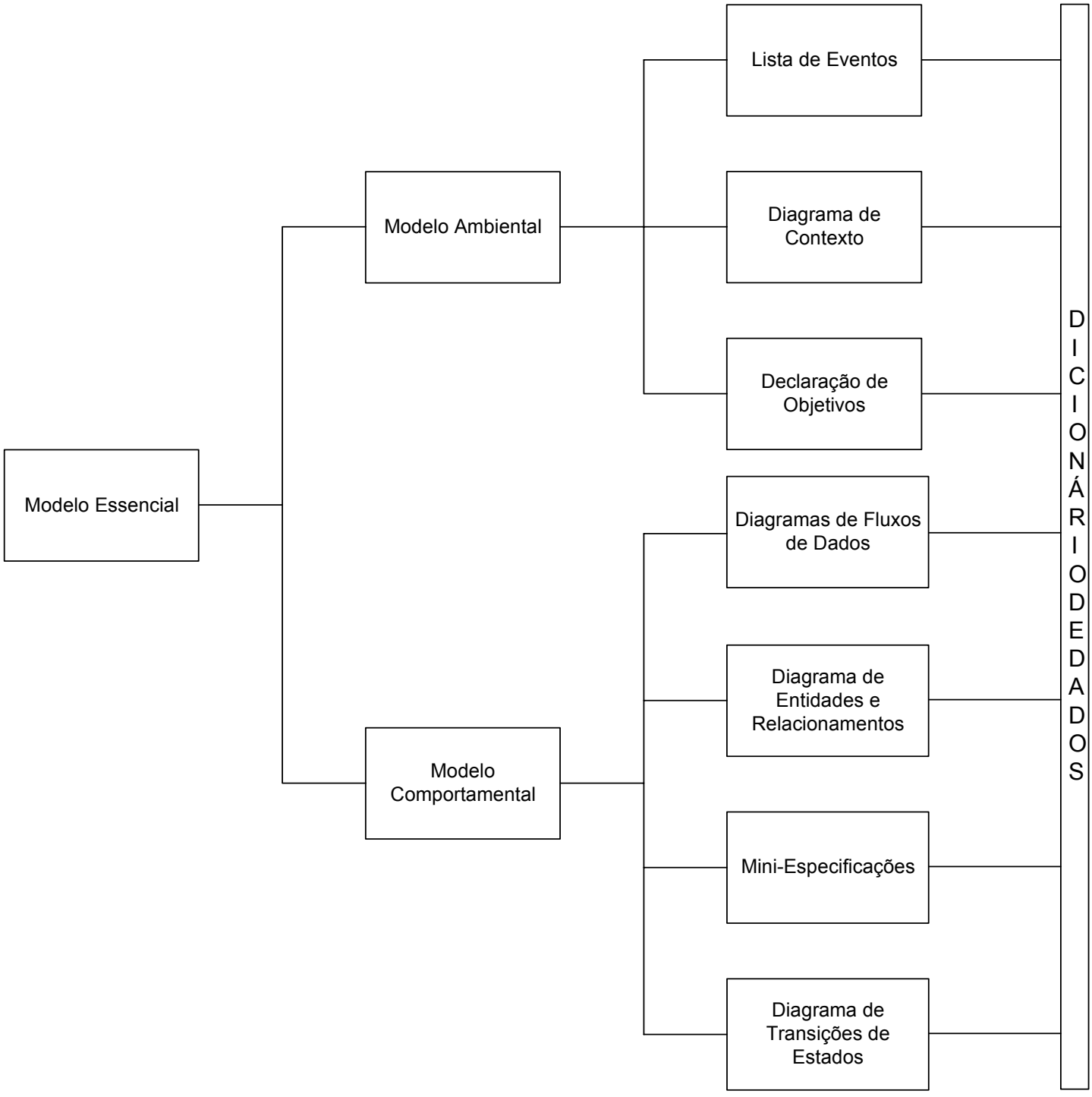
A figura da próxima página apresenta de forma sintética a organização do modelo essencial.

A construção do modelo comportamental pode ser dividida em duas etapas: a construção de um modelo comportamental preliminar e a complementação do modelo comportamental.

3.3.1. A Construção do Modelo Comportamental Preliminar

A construção do modelo comportamental preliminar envolve as seguintes tarefas:

- Construção do DFD Preliminar:
    - Desenha-se um processo para cada evento da lista de eventos. Assim, a quantidade de diagramas deve ser equivalente ao número de eventos na lista.
    - Nomeia-se cada processo de acordo com a resposta que o sistema deve dar ao evento associado. Escolha nomes tão específicos quanto possível. Por exemplo, se um evento for CLIENTE EFETUA PAGAMENTO, um nome adequado para o processo seria ATUALIZAR CONTAS A RECEBER (se for esta a única resposta exigida pelo sistema), em vez de PROCESSAR PAGAMENTO DE CLIENTE (que nada nos informa sobre a natureza da resposta).
    - Desenha-se as entradas e saídas dos processos de modo que este seja capaz de emitir a resposta necessária ao evento. As perguntas que aqui devem ser feitas são: “De que fluxos de dados o processo necessita para cumprir sua função?” e “Que fluxos de dados de saída o processo gera (seja para as entidades externas, seja para os depósitos)?”. Desenha-se os depósitos entre os processos, para que estes possam se comunicar. Lembre-se que, devido a sua natureza assíncrona, os processos só podem se comunicar através de depósitos.
    - Verifica-se a consistência do DFD Preliminar em relação ao diagrama de contexto. As entradas e saídas do diagrama de contexto também devem estar presentes no diagrama preliminar e vice-versa.



Organização do Modelo Essencial.

- Construção do DER Inicial:
  - Também pode ter como ponto de partida a lista de eventos. Os substantivos da lista de eventos muitas vezes serão entidades em um DER. Por exemplo: Para o evento CLIENTE EFETUA PAGAMENTO, identificamos as entidades CLIENTE e PAGAMENTO.
  - Deve-se construir o DFD Preliminar e o DER inicial em paralelo, fazendo-se verificações cruzadas entre os diagramas. Dessa forma, depósitos de dados no DFD podem sugerir entidades no DER e vice-versa.

- Construção do Dicionário de Dados Inicial.

3.3.2. Complementação do Modelo Comportamental

O DFD preliminar desenhado na etapa anterior pode ser muito complexo para ser compreendido por um analista e principalmente por um usuário (imagine um DFD preliminar para uma lista de 40 eventos! Teríamos cerca de 40 processos). Além disso, o DER inicial pode ainda ser muito primitivo, sendo necessário refiná-lo acrescentando-se ou eliminando-se entidades e atributos. A complementação do modelo comportamental envolve as seguintes tarefas:

- Complementação do modelo funcional através na subdivisão do DFD em níveis hierárquicos:

  Usando esta abordagem para a construção de diagramas hierárquicos, adotamos uma estratégia *middle-out* (do meio para fora), onde, a partir dos eventos, estabelecemos processos de resposta aos eventos (meio) para depois agrupá-los em níveis superiores (para cima) e, em seguida, especificá-los e, se necessário, explodi-los (para baixo).

  - Construção do DFD 0. Os processos do DFD 0 são obtidos através da subdivisão do DFD preliminar em níveis ascendentes. Isso quer dizer que precisamos agrupar processos relacionados do DFD preliminar, que corresponderão a uma única bolha no DFD de nível acima (Veja figura 20.1 do Yourdon – pág. 455). Um critério de agrupamento bastante razoável é considerar o grau de coesão e acoplamento entre os processos. As seguintes diretrizes podem ser utilizadas, em conjunto ou em separado, na divisão de um DFD em níveis ascendentes:

    - Cada agrupamento de processos deve envolver respostas estreitamente relacionadas (lembre-se que cada bolha do DFD preliminar tem um nome relativo à resposta a um evento da lista de eventos). Isso habitualmente significa que os processos lidam com dados estreitamente relacionados.
    - Procure oportunidades para ocultar depósitos que apareçam no nível inferior. Assim, se você encontrar um grupo de processos no DFD preliminar relativo ao mesmo depósito, sem que outros processos se refiram a esse depósito, então você deve criar uma bolha em nível mais alto que oculte aquele depósito. Veja figura 20.2 do Yourdon – página 456.
    - Da mesma forma que para a criação dos DFDs, utilize o número de 7 $\pm$ 2 fragmentos de informação (processos ou depósitos) para agregar processos em um processo de nível superior.

  - Subdivisão dos processos do DFD 0 em níveis descendentes. Cada processo de DFD 0 corresponde a um DFD de nível inferior com os processos do DFD preliminar agregados na construção do DFD 0. Mas mesmo os processos de resposta aos eventos da lista de eventos, presentes no DFD preliminar podem não ser processos primitivos, ou seja, podem exigir subdivisões para baixo, em DFDs de níveis inferiores. Isso significa apenas que os processos de resposta aos eventos podem ser demasiadamente complexos para serem descritos em uma única especificação de processos de uma página. Técnicas para isso foram apresentadas na Modelagem Funcional

- Complementação do modelo de dados:

  O DER é refinado e melhorado, identificando-se novas entidades ou entidades desnecessárias e atributos. Mais uma vez, a construção dos DFDs e do DER em paralelo, permite fazer verificações cruzadas entre os diagramas. Dessa forma, depósitos de dados no DFD podem sugerir entidades no DER e vice-versa.

- Complementação do Dicionário de Dados:

  O Dicionário de Dados deve ser complementado, refletindo as mudanças nos diagramas. Ao final, verifique sua consistência em relação aos DFDs, aos DERs e a especificação de processos.

- Especificação dos Processos:

  Utilize técnicas de especificação de processos para especificar todos os processos primitivos dos DFDs. Quando estas especificações estiverem prontas, deverão passar por uma verificação cruzada em relação ao Dicionário de Dados e ao DER.