**Apostila de Análise de Desenvolvimento de Sistemas**

Sumário

# 1 INTRODUÇÃO ...............................................................................................................................2 2 OS SISTEMAS ................................................................................................................................2

2.1 Visão geral..................................................................................................................................................................... 2

2.2 Definição de Sistemas .................................................................................................................................................... 2

2.3 O que é um sistema ....................................................................................................................................................... 3

# 3 SISTEMAS NATURAIS ...............................................................................................................4

3.1 Sistemas feitos pelo Homem .......................................................................................................................................... 4

3.2 Sistemas Automatizados ................................................................................................................................................ 4

3.2.1 Tipos de Sistemas Automatizados ............................................................................................................................... 4

# 4 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE ......................................................5

4.1 Análise de Sistemas ........................................................................................................................................................ 5

4.2 A Equipe de Desenvolvimento ....................................................................................................................................... 5

4.2.1 O Analista de Sistemas ................................................................................................................................................ 5

# 5 FASES DO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO ..............................7

5.1 Diagrama de Fluxo de Dados – DFD ............................................................................................................................. 7

5.1.2 Elementos Básicos do DFD ........................................................................................................................................ 7

5.2 Explosão para um nível de função principal ............................................................................................................... 10

5.3 Explosão das funções principais .................................................................................................................................. 11

# 6 O PARADIGMA DA ORIENTAÇÃO A OBJETO ....................................................................12

6.1 Classes e Objetos ......................................................................................................................................................... 12

6.2 Modelagem de Sistemas ............................................................................................................................................. 12

# 7 A FINGUAGEM DE MODELAGEM UNIFICADA - UML...................................................13

7.1 Fases do Desenvolvimento de um Sistema em UML .................................................................................................. 13 7.2 Visões de um sistema ................................................................................................................................................... 14 **8. DIAGRAMAS DA UML ..............................................................................................................14**

8.1 Diagrama Use-Case (Caso de Uso) .............................................................................................................................. 14

8.2 Diagrama de Classes .................................................................................................................................................... 16

8.3 Diagrama de Sequência ............................................................................................................................................... 18

# 9 MODELAGEM DE DADOS .......................................................................................................18 10 A MODELAGEM DE DADOS ..................................................................................................18

10.1 O Modelo conceitual de dados .................................................................................................................................. 19

10.1.1 Componentes do modelo conceitual de dados ....................................................................................................... 19

# 6 DIAGRAMA DE ENTIDADE-RELACIONAMENTO (DER) ................................................21

**EXEMPLO PRÁTICO DE ANÁLISE ...........................................................................................23**

Locação de Fitas ................................................................................................................................................................ 23

# 1 INTRODUÇÃO

Este material foi desenvolvido para a área (disciplina) de análise e projeto de sistemas. Ele pressupõe um conhecimento razoável dos conceitos e da terminologia de computação. O material foi previsto como apoio ao aluno iniciante à área.

A primeira tarefa do iniciante é adquirir uma percepção do processo básico de analisar e projetar sistemas. Vários livros de texto abrangem o tradicional ciclo de vida do sistema, e muitos são bastante bons; infelizmente, poucos incorporam as metodologias estruturadas. Há muitos livros excelentes sobre as técnicas estruturadas, mas pouco incorporam os princípios básicos da análise e projeto, tão essenciais ao iniciante. Este material introduz ao aluno as metodologias estruturadas dentro do contexto do tradicional ciclo de vida de um sistema.

A análise de sistemas é uma profissão desafiadora e emocionante. Como qualquer outra profissão, a análise de sistemas exige dedicação e muito trabalho.

# 2 OS SISTEMAS

## 2.1 VISÃO GERAL

O fato de você estar lendo este texto indica que já é do seu conhecimento alguma coisa sobre computadores. Alguns leitores serão programadores profissionais ou administradores procurando compreender melhor o processo de análise e projeto de sistemas. O analista de sistemas experiente poderá estar interessado em uma visão geral, concreta, de algumas das novas técnicas utilizadas nesta área. Todos têm algo em comum: sabem o que é um programa. Provavelmente já elaboraram programas; e se não, certamente utilizaram um.

Os programadores e usuários tendem a se concentrar em um único programa. O programador vê um trabalho específico a ser realizado, enquanto o usuário vê um problema específico a ser resolvido. Tente obter uma visão um pouco mais ampla desse programa. Faça-se algumas perguntas: Por que esse programa em particular foi feito? Os programas não surgem por acaso. Obviamente, alguém o desejava, senão ele não teria sido feito; mas, por que esse programa? E por que o programa foi projetado do jeito que foi?

Obviamente, alguém deve ter planejado não só o programa, mas todo o ambiente em que ele se situa. Estamos lidando com algo muito mais amplo do que um programa. Estamos lidando com um sistema.

Este capítulo começa definido o termo “sistema”. Depois, consideramos o processo de análise e projeto de sistemas, enfatizando o trabalho executado pelo analista. Utiliza-se um exemplo de engenharia para ilustrar a necessidade metódica para projetar um sistema grande e complexo. Para o analista, a abordagem estruturada da análise e projeto de sistemas fornece essa metodologia. A chave para esse tipo de análise e projeto estruturado é o ciclo de vida do sistema, de modo que descrevemos sucintamente como esta abordagem orienta o analista através das etapas deste ciclo.

## 2.2 DEFINIÇÃO DE SISTEMAS

**Conceito de Sistemas**

Define-se sistema como um “conjunto de elementos inter-relacionados que possuem características comuns e que podem ser entendidos como um todo”.

Os sistemas são divididos em duas categorias: Naturais e feitos pelo homem.

## 2.3 O QUE é UM SISTEMA



Quantas vezes já nos referimos ou ouvimos a palavra sistema?

 O sistema telefônico ficou mudo!

 O sistema de coleta de lixo está perfeito.

 Chefe, cheguei atrasado porque o sistema de trânsito desta cidade está uma porcaria

Em qualquer um dos casos podemos observar que a palavra sistema está sempre acompanhada de outra que a qualifica. Desta forma, temos o objetivo declarado de um sistema, ou seja, a razão de sua existência. Por exemplo, sistema telefônico, sistema de coleta de lixo, sistema de trânsito, sistema circulatório, sistema educacional, sistema político, sistema médico, sistema nervoso, sistema digestivo, etc.

***Todo e qualquer sistema está inserido em um meio ambiente que o contém, ou seja, tudo o que é externo a um sistema é chamado de seu meio ambiente.***

Em qualquer sistema pode-se encontrar elementos característicos vinculados ao seu fim. No caso do sistema de trânsito, temos veículos, motoristas, pedestres, ruas, guardas, placas, semáforos, etc. Igualmente, pode-se verificar isto em um sistema de controle do acervo de uma biblioteca, onde tem-se: os títulos das obras, os exemplares, os usuários, a localização de cada exemplar, etc.

Observe que estes elementos interagem entre si. Eles se completam e permitem ao sistema atingir seu objetivo.

Podem ser encontradas várias definições para sistema, as quais muitas vezes são extremamente amplas, bastante abrangentes. Nosso trabalho iniciará considerando um sistema um ***conjunto de entidades relacionadas que interagem entre si, buscando atingir um objetivo declarado e outros correlatos***.

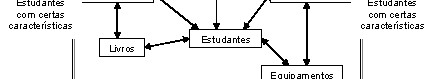
Os elementos aos quais nos referimos são as características do sistema em questão. Esses elementos sempre vão entrar com certas características e, quando saem, possuem novas características.

Exemplo, no Sistema Educacional, encontramos como elementos os estudantes, os professores, os livros, a administração (funcionários) e equipamentos.

Os elementos de um sistema estão relacionados e interagindo entre si com vistas ao objetivo declarado do sistema. Exemplo, professores, alunos, livros, direção, enfim, todos os elementos do sistema educacional buscam atingir juntos o objetivo educacional.

Observe que, dentro de um sistema educacional, certamente se encontrará um sistema de avaliação. Temos então um sistema dentro de outro. Quando isto ocorre, tem-se um subsistema. Portanto, o sistema de avaliação, pelo fato de estar inserido no sistema educacional, é um subsistema deste.

Obs.: Subsistemas também são considerados elementos do sistema onde encontram-se.



# 3 SISTEMAS NATURAIS

A maioria dos sistemas não é feitos pelo homem, e sim encontrados na natureza e com propósito próprio. São basicamente:

* Sistemas físicos: sistemas estrelares, geológicos, etc.
* Sistemas vivos: sistema reprodutor, ingestão, etc.

## 3.1 SISTEMAS FEITOS PELO HOMEM

Alguns são construídos, organizados e mantidos por seres humanos. Entre eles podemos considerar:

* Sistemas sociais, organização de leis, etc.
* Sistemas de transporte: redes rodoviárias, linhas aéreas, etc.
* Sistemas de comunicação: telefone, telex, etc.
* Sistemas de manufatura: fábricas, linhas de montagem, etc.
* Sistemas financeiros: contabilidade, controle de estoques, etc.

## 3.2 SISTEMAS AUTOMATIZADOS

Hoje em dia, muitos desses sistemas não funcionariam sem computadores, mesmo que eles já existiam muito antes do computador. Existem outros que utilizam o computador como componente, mas possui outros componentes não computadorizados ou manuais. Os sistemas feitos pelo homem, que interagem com ou são controlados por um ou mais computadores são denominados sistemas automatizados.

Embora haja muitos tipos diferentes de sistemas automatizados, todo tem componentes em comum:

* Hardware – UCP, Periféricos, Memória, etc.
* Software – Sistemas Operacionais, Programas de Sistemas, Banco de Dados,
* Programas de Controle de Telecomunicação, etc.
* Pessoas – operadores do sistema, que fornecem as entradas e utilizam as saídas.
* Dados – informações que o sistema conserva por um período de tempo.

### 3.2.1 TIPOS DE SISTEMAS AUTOMATIZADOS

a) Sistemas on-line

Usuários interagem com o computador (fornece e recebe dados) por terminais. b) Sistemas de tempo real

Variações dos sistemas on-line. Controlam um ambiente pelo recebimento de dados, seu processamento e apresentação dos resultados com rapidez suficiente para afetar o ambiente naquele momento. Além da velocidade, este sistema interage tanto com pessoas quanto com o ambiente.

1. Sistemas de apoio à decisão

São sistemas tipicamente passivos no sentido de que não funcionam de uma forma regular, em vez disso, são utilizados quando isso se faz necessário. São utilizados pelos diretores para avaliar a missão da empresa, fornecendo informações mais amplas e gerais sobre clientes, comportamento dos competidores, etc.

1. Sistemas baseados no conhecimento

Também conhecidos como sistemas especialistas, são construídos habitualmente para terem a capacidade de explicar as linhas de raciocínio que conduzem a suas decisões.

Alguns podem até mesmo explicar porque rejeitaram certas linhas de raciocínio e escolheram outras. Neste tipo de sistema se armazena dados e regras. Ex.: Sistemas de aprovação de créditos bancários – a decisão é do gerente do banco. e) Sistemas Artificiais

Normalmente são chamados de Sistemas de Informação, justamente porque seu maior objetivo é fornecer, controlar, prover, pesquisar, analisar informações.

Estes sistemas são criados dentro ou para organizações sociais (empresas), considerando dois aspectos: os componentes da empresa e o nível de decisão da empresa.

1. Componentes da empresa: correspondem aos diversos setores que executam as diferentes funções necessárias ao seu

funcionamento ( RH, Vendas,Marketing).

1. Níveis de Decisão: obedecem a hierarquia existente na empresa e são conhecidos como: nível estratégico, tático e

operacional.

# 4 PROCESSO DE DESENVOIVIMENTO DE SOFTWARE

## 4.1 ANÁLISE DE SISTEMAS

A tarefa de construir sistemas de informação é uma das mais complexas e, em última análise, é um processo de solução de problemas.

A Análise de Sistemas consiste nos métodos e técnicas de investigação e especificação da solução de problemas, a partir dos requisitos levantados, para criação e implementação de software em algum meio que o suporte.

## 4.2 A EQUIPE DE DESENVOIVIMENTO

A equipe de desenvolvimento é o conjunto de pessoas responsáveis por construir o software. Dela fazem parte pessoas com diferentes habilidades. Em sistemas de informações tradicionais teremos gerentes de desenvolvimento, analistas, projetistas, programadores, administradores de banco de dados, etc..

Em sistemas mais modernos, como sistemas multimídia e websites, podemos ainda ter profissões novas como artistas e professores.

É importante verificar que as pessoas em uma equipe de desenvolvimento se comunicam de alguma forma. Seguindo a regra de quanto maior o projeto, maior o número de pessoas, muito maior o número de formas em que essas pessoas podem se comunicar.

### 4.2.1 O ANALISTA DE SISTEMAS

Os usuários ou pretensos usuários de computador tem algo em comum: os problemas.

Surge então o profissional Analista de Sistemas, que será o elo entre os usuários e o computador.

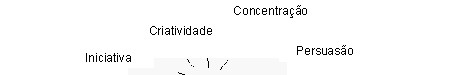
O analista de sistemas deve ser capaz de compreender as disciplinas de engenharia de software e as das atividades da organização. A relação existente entre essas duas áreas e o nível corrente de tecnologia determina a interação entre o exequível e o desejável.

Uma pessoa que atua como analista de sistemas é um bom facilitador e possui habilidades de comunicação acima da média. É fundamental que os profissionais que desempenham este papel tenham conhecimento dos domínios do negócio e da tecnologia.

**Usuário**

Analista

**Programador**



A maior desvantagem em estabelecer uma lista de requisitos como esta é que jamais se encontrará alguém que venha a possuir todos eles. Mesmo assim, não se pode esquecer este panorama, e deve-se tentar conciliar o máximo possível a presença destes requisitos na sua formação profissional.

Para uma boa atuação como Analista de Sistemas, é conveniente observar algumas diretrizes de conduta, que servirão para facilitar seu trabalho:

* Procure ser aceito profissionalmente, do nível mais alto ao nível mais baixo da empresa.
* Tente entender o que o usuário “quer dizer” e não o que “você pensa” que ele quer dizer.
* Escute muito primeiro, fale muito pouco depois!
* Esteja sempre familiarizado com os últimos progressos da tecnologia de informação e compreenda como aplicá-los na sua empresa.
* Seja capaz de explicar conceitos complexos em termos simplificados.
* Não se esconda em jargões da Informática; fale a linguagem da empresa.
* Conheça a área de negócios para a qual desenvolverá sistemas, passando boa parte do seu tempo com o usuário.
* Sugira soluções inovadoras aos requisitos de informação e desenvolva com clareza, analisando sempre a relação custo/benefício e utilizando alternativas viáveis.

# 5 FASES DO DESENVOIVIMENTO DE SISTEMAS DE INFORMAÇÃO

O processo de desenvolvimento de sistemas de informação, também chamado de ciclo de vida, abrange todas as atividades necessárias para definir, desenvolver, testar, operar e manter um sistema. Veja abaixo as fases do ciclo de vida de um Sistema de Informação:

a) Análise

A proposta desta fase é definir e modelar “o que” o sistema irá fazer, independente da tecnologia que será utilizada. Em acréscimo ao modelo do sistema descrevendo os requisitos do usuário, um mais cuidadoso e detalhado conjunto de orçamento e cálculo de custo benefício é preparado, geralmente, ao final da fase de análise. b) Projeto

Nesta fase, o objetivo é de definir a melhor tecnologia, levando-se em conta todas as características do sistema (conforme obtido na fase de análise). A atividade de projeto ocupa-se de partes da especificação aos processadores apropriados e para tarefas apropriadas no interior de cada processador. Em cada tarefa, a atividade de projeto ocupa-se com o desenvolvimento de uma hierarquia apropriada de módulos de programa e interface entre estes módulos para implementar a especificação criada na atividade de análise. Além disso, a fase de projeto ocupa-se com a transformação de modelos de dados de entidade - relacionamento em um projeto de banco de dados.

c) Implementação

Codificar, integrar módulos e criar banco de dados são os objetivos desta fase. Esta é, tipicamente, uma atividade em que o analista de sistemas não está envolvido, embora muitas vezes todas as fases sejam realizadas pela mesma pessoa. d) Implantação

O objetivo desta fase é implantar o sistema nas instalações do usuário. Nesta etapa são prontificados os manuais, os arquivos são carregados, o sistema é instalado e os usuários são devidamente treinados. e) Manutenção

Todo trabalho efetuado após a fase de implantação. A manutenção pode ser de dois tipos: corretiva ou evolutiva. Ela será corretiva, a partir do momento em que o propósito é a correção de erros encontrados no sistema. Ela será evolutiva, a partir do momento em que o propósito é incluir novas funcionalidades no sistema, mesmo depois de implantado.

## 5.1 DIAGRAMA DE FLUXO DE DADOS – DFD

Representa o modelo funcional do sistema.

É utilizado para a representação lógica de processos. O objetivo é descrever graficamente, o que acontece, sem se preocupar em como e quando tais coisas acontecem. Trata-se de uma ferramenta para o modelo funcional do sistema.

Pode ser empregado para a comunicação com pessoal técnico ou não técnico, já que a representação gráfica é de fácil entendimento.

Em resumo o DFD deve explicitar

* Funções do sistema (processos/serviços)
* Interações entre as funções do sistema
* Transformações que o sistema deve realizar
* As fontes de informação
* O destino dos resultados.
* Dados mantidos pelo sistema (dados em repouso)

### 5.1.2 ELEMENTOS BASICOS DO DFD

1. Entidade Externa: pessoa, grupo de pessoas ou subsistema /sistema fora do sistema em estudo que recebem dados do

Sistema e/ou enviam dados para o sistema. As entidades externas funcionam sempre como origem/destino de dados.

1. Fluxo de dados: dados que fluem entre processos, entre processos e arquivos de dados ou ainda entre processos e entidades externas, sem nenhuma especificação temporal (por exemplo ocorrência de processos simultâneos, ou todas as semanas).

1. Depósito de dados: meio de armazenamento de dados para posterior acesso e/ou atualização por um processo.

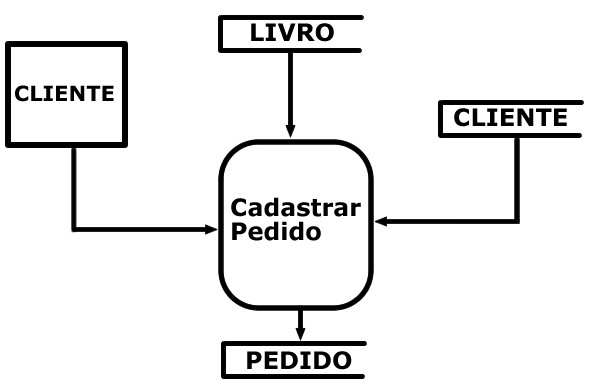
|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | Dn |  | |

1. Processo: recebe dados de entrada e transforma estes dados num fluxo de saída.

nr

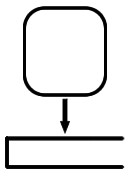
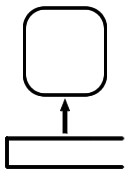
nr

Como exemplo, podemos ter:

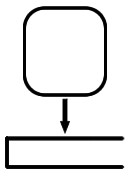
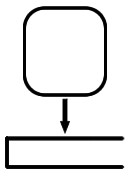


Para representar um fluxo de acesso à memória (depósito de dados), é comum usar:

a) Consultar (Ler) b) Incluir (Gravar)



c) Atualizar d) Excluir



Observe que o processo não é necessariamente um programa. Um único processo poderá representar uma série de programas, um único programa, ou um módulo dentro de um programa; ele poderá até representar um processo manual, tal como a perfuração ou a conferência visual dos dados. Note, também, que o depósito de dados não é o mesmo que um arquivo. Aquele poderá representar o arquivo, uma parcela do arquivo, elementos em uma base de dados, ou mesmo uma parcela de um registro. O depósito de dados poderá residir em disco, fita magnética, memória principal, cartão perfurado, ou qualquer outro meio (inclusive o cérebro humano).

Qual é a diferença entre o depósito de dados e o fluxo de dados? O fluxo de dados representa os dados em movimento; o depósito de dados representa dados em repouso. Em outras palavras, são meramente dois estados da mesma coisa. Voltaremos a esta ideia posteriormente.

Em nome da facilidade de leitura do DFD, é conveniente usar sempre a mesma notação para cada um de seus componentes.

Muitas vezes, ao desenhar um DFD complexo, nos deparamos com a possibilidade de cruzamento de linhas que representam os fluxos de dados, o que dificulta a inteligibilidade do diagrama. Para resolver este problema, a saída costuma ser a repetição, em outro ponto do DFD, de algumas das figuras que representam os seus componentes, da seguinte forma:

1. Para duplicação de entidades externas, costuma-se acrescentar ao símbolo uma linha diagonal para cada repetição

Utilizada, como a seguir:



1. Para duplicação dos depósitos de dados, costuma-se acrescentar ao símbolo uma linha paralela para cada repetição

utilizada, como a seguir:



1. Nos casos em que for inevitável o cruzamento de linhas, uma solução que pode ser usada é o de desenhar uma Ponte (by pass), como a seguir:



1. Não faz sentido duplicar processos. Isto geraria confusão quanto ao entendimento do diagrama.

Para ilustrar as situações acima descritas, observar o DFD a seguir:

Como os fluxogramas lógicos tradicionais, a direção do fluxo é de cima para baixo e da esquerda para a direita. Um bom

|  |  |
| --- | --- |
| D1 | Clientes |

DFD tende a seguir uma convenção semelhante, com os dados deslocando-se de sua origem (na parte superior esquerda) até seu destino (na parte inferior direita), mas as regras são muito menos rígidas. Por exemplo, os dados às vezes fluem de volta até sua fonte. Uma maneira de indicar isso é desenhar uma longa linha de fluxo de um lado do diagrama para o outro. Como alternativa, o símbolo para a origem do dado simplesmente poderá ser repetido – em outras palavras, o mesmo símbolo representando a mesma fonte de dados pode aparecer mais de uma vez no DFD.

**cliente**

1

**cadastrar**

**cliente**

Dados Cliente

Dados Cliente

Válido

**gerar**

**Relatório**

**cliente**

2

**gerênci**

**a**

Dados Cliente

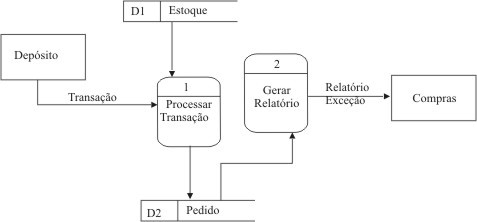
Válido

Lista de Cliente

## 5.2 EXPIOSÃO PARA UM NÍVEL DE FUNÇÃO PRINCIPAL

A figura acima representa talvez a visão mais ampla possível do sistema. Contudo, exceto pelo destaque de origens e destinos de dados, este DFD não é particularmente útil. O passo seguinte é “explodir” o processo dentro de suas partes funcionais. Para fazer isso, dois processos podem ser identificados: *gerar relatório* e *processar transação*. Esses dois processos representam as funções básicas que terão de ser realizadas pelo sistema; elas irão substituir “sistema de reposição”. A explosão de um DFD significa a substituição de um processo de alto nível por seus componentes de nível inferior. Observe abaixo:





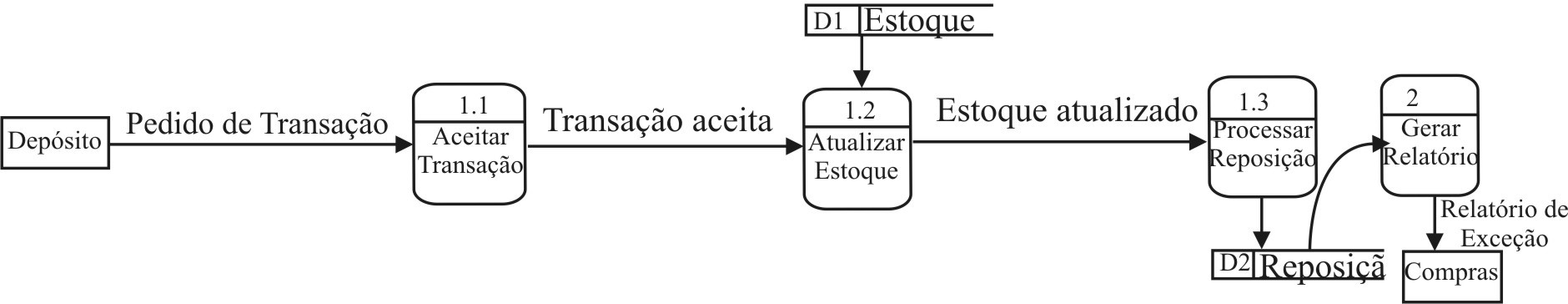
Observe que dois depósitos de dados foram acrescentados ao novo DFD. *Processar transações* precisa de dados do estoque; assim temos o depósito de dados conhecido por *estoque*. Você se lembra da diferença do tempo entre o processamento das transações de estoque e a geração do relatório de reposição? Por causa desta diferença, informações de reposição terão de ser armazenadas; por isso, temos o depósito de dados *reposição*.

A Figura acima ilustra diversas convenções utilizadas para desenhar o DFD. Os processos são numerados para facilidade de consulta. Os depósitos de dados são rotulados com “D” seguido de um número; mais uma vez, esses rótulos são apenas para fins de consulta. Os nomes dos depósitos, origens e destinos de dados são todos escritos em letras maiúsculas, enquanto os nomes dos processos e de fluxos de dados são, exceto a primeira letra, escritos em letras minúsculas. Estas convenções ajudam a tornar o DFD mais fácil de ser seguido.

Os nomes dos vários fluxos de dados são escritos dentro de seu diagrama. Como os processos, origens, destinos e depósitos de dados, eles são derivados da descrição do problema. Considere um desses fluxos – *reposição*. *Reposição* é um fluxo de dados ou um depósito de dados? À primeira vista, distinguir entre depósitos e fluxos poderá parecer confuso. Não se preocupe a respeito. O fluxo de dados são dados em movimento; o depósito de dados são dados em repouso (armazenados). Que elementos são encontrados no depósito de dados? Apenas aqueles elementos que entraram no depósito de dados através de um fluxo de dados. Em outras palavras, o depósito e o fluxo de dados são apenas duas versões diferentes da mesma coisa.

## 5.3 EXPLOSÃO DAS FUNÇÕES PRINCIPAIS

Uma vez identificadas as funções principais e incorporadas ao DFD, o analista pode começar a explodir cada uma dessas funções para um nível inferior de detalhamento. Por exemplo, considere a função processar transações. Logicamente, poderia ser razoável decompor este processo em três passos: aceitar transação, atualizar estoque e processar reposição. Por que estes três passos? Pense a respeito do fluxo lógico dos dados através do sistema. Primeiro, a transação terá de ocorrer e ser aceita. A seguir a transação pode ser processada. Finalmente, uma vez que a fase de processamento tenha determinado que uma reposição é necessária, os dados de reposição podem ser processados. Os três passos terão de ocorrer na ordem prevista. Eles são relativamente independentes, vinculados apenas por um fluxo de dados. Uma decomposição semelhante pode ser imaginada para muitos processos de nível funcional.



*DFD com o processo “processar transações” exploDiDo para um nível inFerior.*

Observe como os processos foram numerados no DFD explodido. Os processos 1.1, 1.2 e 1.3 são partes componentes do que era o processo 1. Se o processo 2 fosse explodido, seus componentes seriam numerados 2.1, 2.2, e assim por diante.

Devemos explodir o segundo processo, gerar relatório? Não. É relativamente fácil de visualizar o que o processo de geração de relatório fará. Embora você possa imaginar diversos meios diferentes de gerar o relatório utilizando uma classificação ou outro mecanismo qualquer de sequenciamento, tais detalhes envolvem especificações físicas e não são apropriados no DFD.

Devemos explodir os processos apresentados na figura anterior para um nível ainda mais baixo? Provavelmente não. Por quê? Como saber quando já desceu ao nível suficiente? Considere o processo 1.1, aceitar transação. Tente imaginar a decomposição desse processo sem pensar como você vai aceitar as transações, ou de onde (fisicamente) essas transações virão. Quando você atingir o ponto em que uma maior subdivisão o levará a pensar em como implementar o processo, será o suficiente.

Qual o detalhamento do Diagrama do Fluxo de Dados? O exemplo apresentado neste módulo é relativamente trivial; o que aconteceria se o DFD contivesse dezenas de processos e depósitos de dados? Esse diagrama seria muito difícil de seguir, prejudicando, assim, uma importante finalidade – a comunicação.

Diversos estudos sugerem que as pessoas encontram dificuldade em seguir um DFD contendo mais de 7+/-2 processos. Esses estudos sugerem uma estratégia. Comece com um diagrama de alto nível. Exploda até o nível funcional. Se a explosão de todas as funções para o seu nível seguinte de detalhamento fazer com que você supere o limite de 9 processos, não exploda o DFD. Ao invés disso, tome cada função, uma de cada vez, e desenvolva um subdiagrama, apresentando apenas a explosão de um único processo. Repita este passo para cada processo. O DFD de nível funcional pode então ser utilizado para fornecer uma visão global lógica do sistema como um todo. Se o usuário ou o gerente quiser saber o que acontece dentro de um processo dado, o subdiagrama apropriado pode ser apresentado.

# 6 O PARADIGMA DA ORIENTAÇÃO A OBJETO

Interessante ao desenvolvimento atual de sistemas de software é o Paradigma da Orientação a objetos. Um paradigma é uma forma de abordar um problema. Considere a história da maça caindo sobre a cabeça de Isaac Newton. Em vez de pensar que somente a maça estava caindo sobre a terra, Newton também considerou a hipótese de o próprio planeta também estar caindo sobre a maçã. Essa outra maneira de abordar o problema pode ser vista como um paradigma.

O paradigma da orientação a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada objeto é responsável por realizar tarefas específicas. È através da interação entre objetos que uma tarefa computacional é realizada.

Pode-se concluir que a orientação a objetos, como técnica para modelagem de sistemas, diminui a diferença semântica entre a realidade sendo modelada e os modelos construídos.

## 6.1 CLASSES E OBJETOS

Focamos alguns princípios da orientação a objetos:

1. Qualquer coisa é um objeto;
2. Objetos realizam tarefas através da requisição de serviços a outros objetos;
3. Cada objeto pertence a uma determinada classe. Uma classe agrupa objetos similares;
4. A classe é um repositório para comportamento associado ao objeto;
5. Classes são organizadas em hierarquias.

Uma classe é um modelo para objetos. Normalmente você não usaria as classes diretamente em seu programa. Ao invés disso, você as usaria apenas para instanciar novos objetos.

Imagine uma classe como uma planta de uma cada: você não pode morar na planta da casa, mas pode usá-la para criar quantas casas quiser.

## 6.2 MODELAGEM DE SISTEMAS

Na década de 1990 surgiram várias propostas de técnicas para modelagem de sistemas segundo o paradigma orientado a objetos. Percebeu-se a necessidade da existência de uma linguagem que viesse a se tornar um padrão para a modelagem de sistemas, que fosse aceita e utilizada amplamente pela indústria e pelos ambientes acadêmicos. E os esforços deram surgimento na definição da UML ( Unified Modeling Language ) em 1996 como a melhor candidata para ser a linguagem “unificadora” de notações , diagramas e formas de representação existentes em diferentes técnicas.

# 7 A LINGUAGEM DE MODELAGEM UNIFICADA - UML

A UML é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.

A UML é uma linguagem visual para modelar sistemas orientados a objetos. Isso quer dizer que a UML é uma linguagem constituída de elementos gráficos (visuais) utilizados na modelagem que permitem representar os conceitos do paradigma da orientação a objetos. Através de elementos gráficos definidos nesta linguagem pode-se construir diagramas que representam diversas perspectivas de um sistema.

## 7.1 FASES DO DESENVOFVIMENTO DE UM SISTEMA EM UMF

Existem cinco fases no desenvolvimento de sistemas de software: análise de requisitos, análise, design (projeto), programação e testes. Estas cinco fases não devem ser executadas na ordem descrita acima, mas concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente de forma que o resultado global gere um produto de alta qualidade e performance. A seguir falaremos sobre cada fase do desenvolvimento de um sistema em UML:

a) Análise de Requisitos

Esta fase captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através do uso de funções chamadas “use-cases”. Através do desenvolvimento de “use-case”, as entidades externas ao sistema (em UML chamados de “atores externos”) que interagem e possuem interesse no sistema são modelados entre as funções que eles requerem, funções estas chamadas de “use-cases”. Os atores externos e os “use-cases” são modelados com relacionamentos que possuem comunicação associativa entre eles ou são desmembrados em hierarquia. Cada “use-case” modelado é descrito através de um texto, e este especifica os requerimentos do ator externo que utilizará este “use-case”. O diagrama de “usecases” mostrará o que os atores externos, ou seja, os usuários do futuro sistema deverão esperar do aplicativo, conhecendo toda sua funcionalidade sem importar como esta será implementada. A análise de requisitos também pode ser desenvolvida baseada em processos de negócios, e não apenas para sistemas de software. b) Análise

A fase de análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no Diagrama de Classe. As colaborações entre classes também são mostradas neste diagrama para desenvolver os “use-cases” modelados anteriormente, estas colaborações são criadas através de modelos dinâmicos em UML. Na análise, só serão modeladas classes que pertençam ao domínio principal do problema do software, ou seja, classes técnicas que gerenciem banco de dados, interface, comunicação, concorrência e outros não estarão presentes neste diagrama. c) Design (Projeto)

Na fase de design, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infraestrutura técnica: a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, dentre outros. As classes do domínio do problema modeladas na fase de análise são mescladas nessa nova infraestrutura técnica tornando possível alterar tanto o domínio do problema quanto a infraestrutura. O design resulta no detalhamento das especificações para a fase de programação do sistema.

d) Programação

Na fase de programação, as classes provenientes do design são convertidas para o código da linguagem orientada a objetos escolhida (a utilização de linguagens procedurais é extremamente não recomendada). Dependendo da capacidade da linguagem usada, essa conversão pode ser uma tarefa fácil ou muito complicada. No momento da criação de modelos de análise e design em UML, é melhor evitar traduzi-los mentalmente em código. Nas fases anteriores, os modelos criados são o significado do entendimento e da estrutura do sistema, então, no momento da geração do código onde o analista conclua antecipadamente sobre modificações em seu conteúdo, seus modelos não estarão mais demonstrando o real perfil do sistema. A programação é uma fase separada e distinta onde os modelos criados são convertidos em código. e) Testes

Um sistema normalmente é rodado em testes de unidade, integração e aceitação. Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador. Os testes de integração são aplicados já usando as classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos. Os testes de aceitação observam o sistema como uma “ caixa preta” e verificam se o sistema está funcionando como o especificado nos primeiros diagramas de “use-cases”.

O sistema será testado pelo usuário final e verificará se os resultados mostrados estão realmente de acordo com as intenções do usuário final.

## 7.2 VISÕES DE UM SISTEMA

O desenvolvimento de um software complexo demanda que seu desenvolvedor tenham a possibilidade de examinar e estudar esse sistema a partir de diversas perspectivas. Estas perspectivas são definidas como visões. Cada visão enfatiza aspectos diferentes do sistema.

* **Visão de casos de Uso:** descreve o sistema de um ponto de vista externo como um conjunto de interações entre o sistema e os agentes externos ao sistema.
* **Visão de Projeto:** enfatiza as características do sistema que dão suporte, tanto estrutural quanto comportamental, às funcionalidades externamente visíveis do sistema.
* **Visão de Implementação:** abrange o gerenciamento de versões do sistema, construídas através do agrupamento de módulos (componentes) e subsistemas.
* **Visão de Implantação:** corresponde à distribuição física do sistema em seus subsistemas e à conexão entre essas partes.
* **Visão de Processo:** enfatiza as características de concorrência (paralelismo), sincronização e desempenho do sistema.

# 8. DIAGRAMAS DA UML

O objetivo dos diagramas é apresentar múltiplas visões do sistema sendo que este conjunto de múltiplas visões é chamado de modelo. Podemos dizer que um modelo UML pode ser visto como um conjunto de diagramas que podem ser examinados e modificados a fim de compreender e desenvolver um sistema de software.

Um modelo UML descreve o que o sistema fará, mas não diz nada como implementar o sistema. Um diagrama provê uma parcial representação do sistema. Ele ajuda a compreender a arquitetura do sistema em desenvolvimento. Vejamos s seguir os diagramas da UML Estáticos:

* Diagramas de Use Cases
* Diagramas de Classes
* Diagramas de Pacotes

**Dinâmicos:**

* Diagramas de Interação
* Diagramas de Sequência
* Diagramas de Colaboração
* Diagramas de Estado (Statechart)
* Diagramas de Atividade

## 8.1 DIAGRAMA USE-CASE (CASO DE USO)

O Diagrama de Use Case tem o objetivo de auxiliar a comunicação entre os analistas e o cliente. Ele descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário, ou seja, é uma técnica usada para descrever e definir os requisitos funcionais do sistema.

O cliente deve ver no diagrama de Use Cases as principais funcionalidades de seu sistema.

O diagrama de Use Cases é representado por:

* Atores;
* Use cases;
* Relacionamentos entre estes elementos.

Estes relacionamentos podem ser:

**>** associações entre atores e use cases;

**>** generalizações entre os atores;

**>** generalizações, expendes e includes entre os use cases.

**ATORES**

Um ator é representado por um boneco e um rótulo com o nome do ator. Um ator é um usuário do sistema, que pode ser um usuário humano ou um outro sistema computacional.

**Ator**

**USE CASE**

Um use case é representado por uma elipse e um rótulo com o nome do use case. Um use case é uma funcionalidade do sistema.

### RELACIONAMENTOS

Ajudam a descrever os use cases.

1. Entre um ator e um use case: a. Associação

Define uma

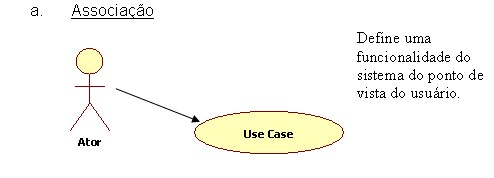
Funcionalidade do sistema do ponto de vista do usuário.

**Ator**

**Use Case**

1. Entre atores:

a. Generalização



3) Entre use cases: a. Include

Um relacionamento include de um use case A para um use case B indica que B é essencial para o comportamento de A. b. Expendes

Um relacionamento extend de um use case A para um use case B indica que o use case A pode ser acrescentado para descrever o comportamento de B (não é essencial).

A extensão é inserida no ponto de extensão do use case B.

Ponto de extensão em um use case é uma indicação de que outros use cases poderão ser adicionados a ele. Quando o use case for invocado, ele verificará se suas extensões devem ou não serem invocadas.

c. Generalização ou Especialização (é um)

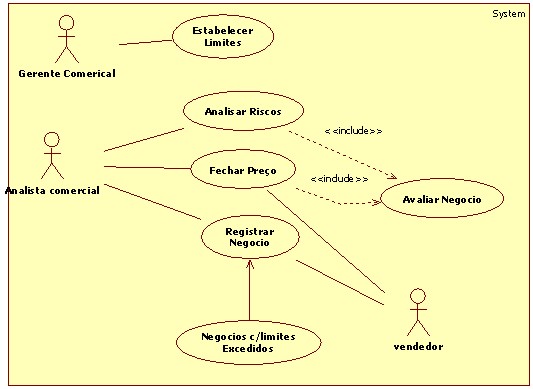
Use case B é um use case A (A é uma generalização de B, ou B é uma especialização de A).

Um relacionamento entre um use case genérico para um mais específico, que herda todas as características de seu pai.

**SISTEMA**

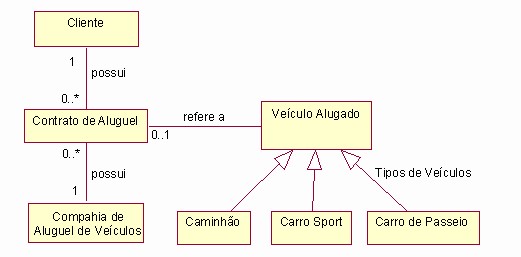
**limites do sistema:** representado por um retângulo envolvendo os use cases que compõem o sistema.

**Nome do sistema:** Localizado dentro do retângulo.



## 8.2 DIAGRAMA DE CLASSES

O diagrama de classes demonstra a estrutura estática das classes de um sistema onde estas representam as “coisas” que são gerenciadas pela aplicação modelada. Classes podem se relacionar com outras através de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares).



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Nome classe** | | +Atributos | | +métodos() | |  | |  | | --- | | **Cliente** | | +nome: int -idade: string | | +criar() +apagar() | |

Notação:

1. **Nome da classe:** Um identificador para a classe, que permite referenciá-la posteriormente -- por exemplo, no momento da criação de um objeto.
2. **Atributos:** O conjunto de propriedades da classe. Para cada propriedade, especifica-se:
3. Nome: um identificador para o atributo.
4. Tipo: o tipo do atributo (inteiro, real, caráter, outra classe, etc.)
5. Valor default: opcionalmente, pode-se especificar um valor inicial para o atributo.
6. Visibilidade: opcionalmente, pode-se especificar o quanto acessível é um atributo de um objeto a partir de outros

objetos. Valores possíveis são:

- (privativo), nenhuma visibilidade externa;

+ (público), visibilidade externa total;

# (protegido), visibilidade externa limitada.

c) **Métodos (Operações):** também são exibidos com pelo menos seu nome, e podem também mostrar seus parâmetros

e valores de retorno. Operações podem, como os Atributos, mostras sua visibilidade:

+ indica operações públicas

# indica operações protegidas

- indica operações privadas

**Conta**

-

agencia: int

-

contacorrente: char

+

()

depositar

()

saldo

+

**ContaPoupanca**

DiaDeposito: int

-

+

VerLucro

()

**Cliente**

-

CPF: int

-

nome: string

-

telefone: char

+

MostrarCPF

()

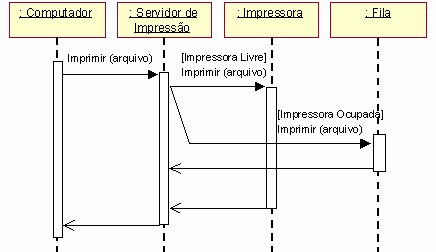
VerSaldo

()

+

## 8.3 DIAGRAMA DE SEQUÊNCIA

Diagramas de Sequência mostram a troca de mensagens (isto é chamada de método) entre diversos Objetos, numa situação específica e delimitada no tempo. Objetos são instâncias de classes. Diagramas de Sequência colocam ênfase especial na ordem e nos momentos nos quais mensagens para os objetos são enviadas.



# 9 MODELAGEM DE DADOS

O modelo de entidade-relacionamento (MER) é baseado na percepção do mundo real que consiste em um conjunto de objetos básicos chamados entidades e nos relacionamentos entre estes objetos.

# 10 A MODELAGEM DE DADOS

Trata-se de parte do trabalho do analista de sistemas, cujo propósito é buscar especificar, a partir dos fatos essenciais que estejam associados ao domínio de conhecimento analisado, a perspectiva dos dados, permitindo organizá-los em estruturas bem definidas, estabelecer as regras de dependência e restrições entre eles, produzindo um modelo expresso por uma representação, ao mesmo tempo, descritiva e diagramática.

Na literatura de Informática, de um modo geral, os termos “dado” e informação” costumam ser utilizados como sinônimos, porém, trata-se de coisas distintas, cada qual com seu conceito.

**Dado = Atributo + Valor**

Os ***dados*** podem ser vistos como uma coleção de símbolos organizados intencionalmente para representar uma parte da realidade de que estivemos tratando.

Podemos definir ***informação*** como sendo “conhecimento novo”. Por ser conhecimento, a informação existe apenas na mente das pessoas, ou seja, a informação só pode ser assim chamada se representar acréscimo de conhecimento para alguém. Os dados podem ser usados para representar informações, aumentando, desta forma, o conhecimento de alguém a respeito de algum assunto ou situação.

A modelagem de dados, começa no momento em que um analista de sistemas, define algum depósito de dados no DFD particionado por evento. Tal fato significa que o analista, ao examinar o domínio de seu problema no mundo real, interpretou que, para aquele determinado evento, haveria a necessidade de se armazenar alguma informação sobre algo. Esta interpretação do analista é chamada de visão a nível conceitual, cuja intenção é espelhar a realidade. Deste fato decorre um processo a nível de dados conhecido por *abstração de dados*, ou seja, se tenho um usuário no sistema, devo verificar se é necessário armazenar dados sobre ele. Em caso afirmativo, pergunto “quais dados sobre este usuário devo armazenar? ” A resposta é: “certamente aqueles que são relevantes para o seu sistema”.

Esta ideia conceitual, ainda que preliminar, sobre os dados a serem armazenados, segundo uma visão interpretada do mundo real, é a chamada *abstração de dados*.

## 10.1 O MODELO CONCEITUAL DE DADOS

O valor de um modelo conceitual de dados é tanto maior quanto sua aderência à realidade do mundo que ele se propõe a representar. Antes de pensarmos em implementação, é necessário que tenhamos uma descrição da realidade tão fielmente retratada quanto possível. Uma técnica de modelagem de dados deve procurar capturar os conceitos do ambiente a ser modelado, bem como expressar-se em uma linguagem (notação) que facilite a percepção do usuário a respeito da organização da realidade que o cerca. Além disso, o método proposto deve permitir também que se possa estabelecer agrupamentos das classes de elementos do mundo real sob diversas formas de agregação.

Podemos, então, estabelecer que o ***método de modelagem conceitual de dados*** satisfaça, entre outros, aos seguintes ***requisitos***:

1. Expandir a habilidade do analista no reconhecimento dos requisitos de informação;
2. Capacidade de capturar ao máximo os conceitos do mundo real durante o próprio processo de modelagem;
3. Facilidade de entendimento, através de uma representação pragmática, que possa, inclusive, ser entendida por usuários não-técnicos de Informática.

### 10.1.1 COMPONENTES DO MODELO CONCEITUAL DE DADOS

Os quatro componentes primitivos do modelo, que representam o mundo real, são: entidades, relacionamentos, atributos e domínios.

 **Entidade**

A denominação usada para referir-se ao primeiro conceito, embora seja um tanto esquisita, já se consolidou pelo seu amplo uso na área de modelagem de dados e de projeto de banco de dados. Segundo o autor J. Ullman, uma ***entidade*** é algo que existe e que possui características que a tornam distinguíveis.

No dicionário de Aurélio Buarque de Holanda, encontramos ainda as seguintes definições, que podem nos ajudar a entender do que estamos falando:

“*Entidade:*

1. Aquilo que constitui a essência de uma coisa, existência, individualidade, ente, ser;
2. Tudo quanto existe ou pode existir.”

Assim, uma ***entidade*** é alguma coisa que desempenha um papel específico no sistema que está sendo modelado. É algo sobre o qual desejamos guardar informações. A existência e a identificação de uma entidade dependem inteiramente do contexto em que ela estiver inserida. Por exemplo, em um sistema acadêmico, uma pessoa (que seria uma entidade) pode ter o papel de aluno ou professor. Esta mesma pessoa, em um sistema de arrecadação de atributos, será, digamos, um contribuinte, enquanto em um sistema de competições esportivas, ela poderá ser um atleta. Como vemos, nenhuma entidade pode ser identificada independentemente de um universo de interesse.

Uma entidade pode ser:

1. Um objeto real, como um livro, uma máquina, um lugar, um avião;
2. Uma pessoa, como um empregado, um contribuinte, um aluno, um cidadão;
3. Um conceito abstrato, como um curso, uma cor, uma disciplina;
4. Um acontecimento, isto é, uma situação em que algo está ocorrendo ou está planejado, como o fornecimento de uma

encomenda, a inscrição de um aluno em um curso, uma reunião, um casamento, etc.

Um grupo de entidades possuindo os mesmos atributos forma um “conjunto” de entidades. Um conjunto de entidades deve ser encarada como uma reunião (família, conjunto) de entidades (elementos) com características comuns. Assim, a classe de entidades “pessoa” teria as entidades João, Virgínia, José, Teresa, etc.

Exemplos de entidades:

* O conjunto de todos os empregados;
* O conjunto de todos os fornecedores; • o conjunto de todos os departamentos;
* O conjunto de todos os alunos.
* **Relacionamento**

Dentro do contexto de um sistema, os conjuntos de entidades relacionam-se entre si. Definiremos ***relacionamento*** como uma associação entre duas entidades. Por exemplo, o relacionamento que expressa todas as encomendas feitas a um fornecedor pode ser expresso:

Fornecedor *fornece* encomenda ou

Encomenda *é fornecida por* fornecedor,

Onde Fornecedor e Encomenda são conjuntos de entidades. Por outro lado, “*fornece*” e “*é fornecida por*” representam o relacionamento entre as duas entidades.

* **Atributo**

Conforme já dissemos, nosso interesse em identificar entidades está no fato de desejarmos armazenar dados a seu respeito. Tais dados representam características destas entidades. Assim, *nº-de-matrícula*, *nome*, *data-de-nascimento* e *endereço* podem ser, por exemplo, características da entidade *empregado*. As características ou propriedades das entidades recebem a denominação de ***atributo***. Atributo pode ser entendido como sendo aquilo que o intelecto percebe a respeito da entidade como constituindo a essência dela, isto é, a determinação de uma propriedade essencial da entidade. Desta forma, cada empregado é uma entidade que possui os atributos mencionados. Todas as entidades de um determinado conjunto possuem os *mesmos* atributos.

A cada atributo de uma entidade é associado um ***domínio*** de valores. Estes domínios podem ser um conjunto de números inteiros, números reais, cadeia de caracteres ou qualquer outro tipo de valores. Por exemplo, as entidades do conjunto de entidades *fornecedor* podem ter atributos como:

* Razão social (uma cadeia de caracteres);
* Capital social (um número real);
* Data de fundação da empresa (uma data), etc.

O domínio é o conjunto de valores válidos para um determinado atributo. O atributo “*nome-do-empregado*” tem como domínio (João, José, Maria, ...). Isto é, João, José, Maria, etc. são valores que podem ser atribuídos ao nome do empregado. O atributo *cor-do-carro* tem como domínio (azul, verde, amarelo, ...), ou seja, azul, verde, amarelo, etc. são valores possíveis para a cor do carro.

Ele foi desenvolvido para facilitar o projeto de banco de dados, permitindo a especificação de um esquema de “negócio”, onde tal esquema representa a estrutura lógica geral do banco de dados.

# 6 DIAGRAMA DE ENTIDADE-RELACIONAMENTO (DER)

Integrantes do Modelo:

* Entidades: Conjuntos de “coisas” que possuem características próprias.
* Atributos: Representam as características que qualificam uma Entidade.
* Relacionamentos: Condições que permitem o estabelecimento do nível de associação entre Entidades.

**Entidades:**

O conceito fundamental da abordagem entidade-relacionamento (E-R) é o conceito de entidade.

* Conjunto de objetos da realidade modelada sobre os quais deseja-se colecionar dados no banco de dados
* Pode ser concreta (pessoa, disco,) ou abstrata (curso, conceito, circulação, ...) • uma entidade representa um conjunto de objetos que se deseja guardar dados Exemplo:
* Sistema bancário – as entidades podem ser: clientes, contas correntes, cheques, agências.
* Cliente – representa o conjunto de clientes que se deseja manter dados no banco de dados

|  |  |  |
| --- | --- | --- |
| **Aluno** |  | **Nota fiscal** |

**Atributos:**

Dado que é associado a cada ocorrência de uma entidade ou um relacionamento.

* São informações importantes que caracterizam uma entidade ou relacionamento.
* Os atributos de uma entidade são independentes de todas as demais entidades.

Exemplo:

 Cliente – cada ocorrência de cliente terá associado exatamente os seus atributos (nome, CPF, telefone, endereço)

**Aluno**

matricul

a

Nome

**Chave**

É um ou mais atributos que permite identificar unicamente uma entidade no conjunto de entidades.

**Relacionamento**

Conjunto de associações entre entidades.

* Um conjunto de relacionamentos é uma coleção de ocorrências das entidades relacionadas.
* A função que uma entidade exerce em um relacionamento é chamada de papel, normalmente implícito, mas muito esclarecedor.
* Também pode ter atributos descritivos (por exemplo: data, hora, etc.)
* A ocorrência de um relacionamento particular dentro de um conjunto de relacionamentos de um mesmo tipo é chamada de instância do relacionamento.

Exemplo:

Suponha o relacionamento “lotação” entre as entidades Departamento e Pessoa.

Este exemplo expressa que o BD armazenará dados sobre:

* Um conjunto de objetos classificados como pessoa entidade Pessoa;
* Um conjunto de objetos classificados como departamentos entidade Departamento;
* Um conjunto de associações entre cada pessoa e um departamento; relacionamento lotação

**lotação**

**Cardinalidade**

É uma restrição de mapeamento que expressa o número de entidades as quais outras entidades podem ser associadas via um conjunto de relacionamentos.

**Um para um (1:1):** uma entidade de A está associada a uma única entidade de B, e uma entidade de B está associada a uma única entidade de A.

**Um para muitos (1:N):** uma entidade de A está associada a qualquer quantidade da entidade de B, e uma entidade de B esta associada somente a uma única entidade de A.

**Muitos para um (N:1):** uma entidade de A está associada a uma única entidade de B, e uma única entidade de B pode estar associada a qualquer quantidade de entidades de A.

**Muitos para muitos (N:M):** uma entidade de A está associada a qualquer quantidade de entidades de B, e uma entidade de B esta associada a qualquer quantidade de entidades de A.

Observe:

a

a1

a2

a3

b1

b2

b3

a

b

a1

a2

a3

b1

b2

b3

b

4

b

Relacionamento um para um

a1

a2

a3

a4

b1

b2

b3

a

b

Relacionamento muitos para um Relacionamento um para muitos

a1

a2

a3

a4

b1

b2

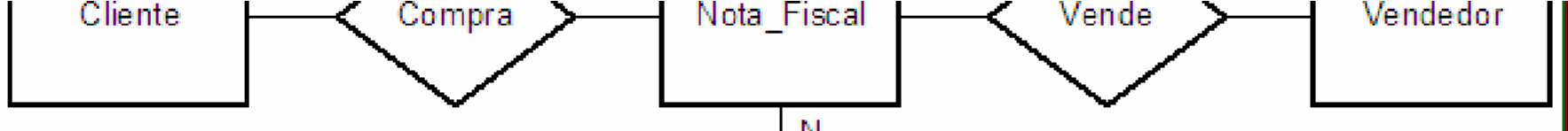
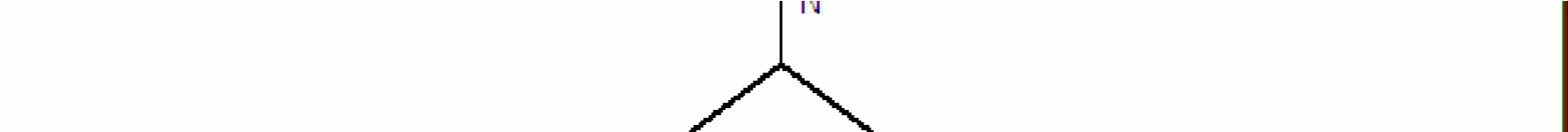
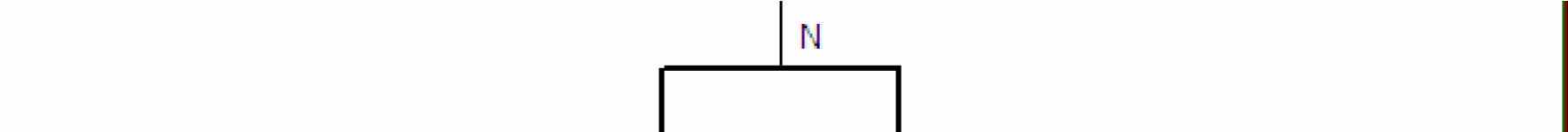
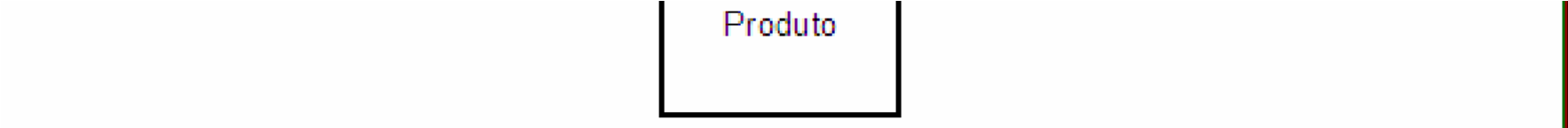
b3

b4

a

b

Relacionamento muitos para muitos



# EXEMPLO PRÁTICO DE ANÁLISE

## LOCAÇÃO DE FITAS

Desenvolva um Diagrama de Casos de Uso para um sistema de vídeo locadora equivalente ao módulo de locação de fitas de filmes de acordo com as seguintes afirmações:

1. Ao realizar uma locação, o sócio deve primeiro informar seu código para que o atendente possa verificar se este se encontra cadastrado. Se o sócio não estiver cadastrado, então a locação deverá ser recusada e o sócio será informado de como proceder para se cadastrar. Caso esteja cadastrado, o atendente deve verificar se o sócio em questão já devolveu todas as locações feitas anteriormente, se não o tiver feito, a locação deverá ser recusada.
2. Caso o sócio tenha quitado todas as locações anteriores, então este deverá informar os números das cópias dos filmes

que deseja locar. Em seguida o atendente registrará a locação e fornecerá as cópias em questão para o sócio.

1. É responsabilidade de o atendente realizar a manutenção dos filmes e de suas respectivas cópias, registrando os novos filmes adquiridos pela locadora, por exemplo.

**CadastrarSocio**

**Cliente**

**Atendente**

**RealizarLocacao**

**ManterFilmes**

Desenvolva um diagrama de classes para um sistema de vídeo locadora equivalente ao módulo de locação de fitas de filmes de acordo com as seguintes afirmações:

1. Um filme tem obrigatoriedade de pelo menos uma cópia, mas pode possuir diversas delas, porém uma cópia refere-se

exclusivamente a um determinado filme.

1. Um sócio pode realizar muitas locações enquanto permanecer sócio da locadora, mas uma locação refere-se unicamente

a um determinado sócio.

c) Cada locação deve obrigatoriamente referenciar-se ao menos a uma cópia de um filme, podendo referenciar-se a

muitas cópias, no entanto uma mesma cópia pode ter sido locada diversas vezes, em épocas diferentes obviamente.

**Filme**

Titulo: char

[60]

+

+

Duracao: Time

**Copia**

+

NroCopia: int

DataCopia: Date

+

**ItemLocacao**

+

Preco

+

Registrar(): int

**Locacao**

DataLoc: Date

+

DataDev: Date

+

+

Valor: float

Situacao: int

+

+

Locar(): int

VerPendencias(): int

+

**Socio**

[40]

Nome: char

+

+

Endereco: char

[40]

+

Telefone: char

[15]

+

Consulta(nome: char): int

Possui

1..\*

Compoe

0..\*

Contem

1..\*

Realiza

1..\*

Desenvolva o diagrama de sequência para um sistema de vídeo locadora de acordo com os fatos já descritos anteriormente e nos seguintes fatos complementares:

1. Primeiramente o atendente deve verificar se o sócio está cadastrado, se este não estiver, a locação deve ser recusada.
2. Em seguida deve verificar se o sócio possui alguma locação pendente, caso que também recusará o empréstimo.
3. Se o sócio existir e não tiver locações pendentes, então a locação deverá ser registrada e as cópias emprestadas ao

sócio.

1. Durante o registro da locação, deverão ser registrados também todos os ítens da locação.

: Atendente

Interface do Sistema

boundary

>>

<<

s1 : Socio

: Locacao

loc1 : Locacao

: ItemLocacao

()

: Verificar sócio

1

()

2

: Consulta

()

: VerPendencias

3

: falso

4

5

: Verdadeiro

: Sócio válido, Não há pendências

6

7

: Registrar Locação

()

8

: Locar

()

9

\*[para cada item] : Registrar

()

10

: itens registrados

11

: verdadeiro

12

: Locação registrada com sucesso