MODELAGEM DE SISTEMAS

ORIENTAÇÃO A OBJETOS E UML

Olá!

Bom dia!

A modelagem de sistemas consiste no desenvolvimento de modelos (diagramas) que representem o mundo real, sob a perspectiva do desenvolvimento de sistemas e está intimamente relacionada a dois fatores: ao paradigma de desenvolvimento e ao processo de desenvolvimento em uso.

No paradigma orientado a objetos, o enfoque está na identificação de classes e seus respectivos objetos que interagem para a solução do problema.

Dentro da perspectiva do desenvolvimento orientado a objetos, surge a UML (Unified Modelling Language), uma linguagem unificada de modelagem, que permite aos desenvolvedores construírem diagramas sob diferentes perspectivas. Nesta aula, vamos conhecer esses conceitos.

Bons estudos!

Objetivos

•Fundamentar as principais características e os pilares do paradigma orientado a objetos;

•Discutir os principais conceitos inerentes às etapas de análise e ao projeto orientado a objetos;

•Reconhecer os diagramas da UML.

Desenvolvimento de sistemas

O **desenvolvimento de sistemas** é um **processo intelectual e progressivo**, em que os profissionais envolvidos adquirem mais **conhecimento do sistema** à medida que **avançam no entendimento da realidade** em que o sistema está inserido.

Tomemos como exemplo a **criação de um sistema que gerencie as atividades de um estacionamento de veículos**:

No **início do processo**, é preciso conversar com os profissionais envolvidos no negócio em questão para **compreender a realidade** e o **funcionamento do mesmo**. No primeiro dia, nada conhecemos do negócio estacionamento, no quinto dia de estudo, por exemplo, o conhecimento da realidade é mais apurado.

Representação da realidade

Para **representar a realidade** e entender o que se passa no contexto do sistema a ser construído, precisamos traduzir a realidade em modelos.

No contexto de desenvolvimento de sistemas, o que são **modelos**?

Nada mais são que **diagramas gráficos** que representam a realidade e ajudam os desenvolvedores a **compreendê-la.** Portanto, **modelar um sistema** consiste em **criar um conjunto de modelos** sob a forma de **diagramas**, que representam a **estrutura** e o **comportamento de um sistema**.

Saiba mais

Podemos citar como exemplo o caso de uma construtora que, quando idealiza um empreendimento, constrói uma maquete que representa a realidade, ou seja, ela constrói um modelo da realidade. Pela maquete, tanto a equipe de construção como os clientes têm a noção do que será o empreendimento (disposição no terreno, área comum etc.), diminuindo, consideravelmente, a abstração e aproximando-o da realidade. Da mesma forma, um diagrama gráfico representa o sistema, sob determinada visão.

Mas a maquete não é o único modelo usado para uma construção civil. São necessárias, também, diferentes plantas que descrevam as características de cada etapa da construção, tais como planta baixa, planta hidráulica, planta elétrica e outras. Analogamente a essas plantas, temos, no contexto da modelagem de sistemas, outros diagramas, que ajudarão os técnicos a especificar como o sistema deve ser construído.

Conceitos do Paradigma Orientado a Objetos

Podemos dizer que a representação da realidade ocorre por meio de **paradigmas**.

Paradigma é uma **forma de abordar um problema**.

Até o **final dos anos 1980**, prevaleceu o **paradigma imperativo ou estruturado**, caracterizado pelo **uso das técnicas de Análise Estruturada e Análise Essencial**, que foram eficientes para sistemas de pequeno porte.

À medida que os sistemas tornaram-se mais complexos e robustos, uma nova abordagem se fez necessária para estudar, entender e modelar o sistema a ser desenvolvido. Surgia o **paradigma Orientado a Objetos**, também chamado **paradigma OO**.

Na modelagem sob o enfoque da orientação a objetos (OO), a tarefa principal passa a ser a **identificação dos objetos do mundo real envolvidos no contexto do sistema e a relação entre eles**. Ao identificar os objetos, são identificados também os **dados (atributos)** e as **funcionalidades (funções)** inerentes àquele objeto, conforme ilustrado na equação abaixo.

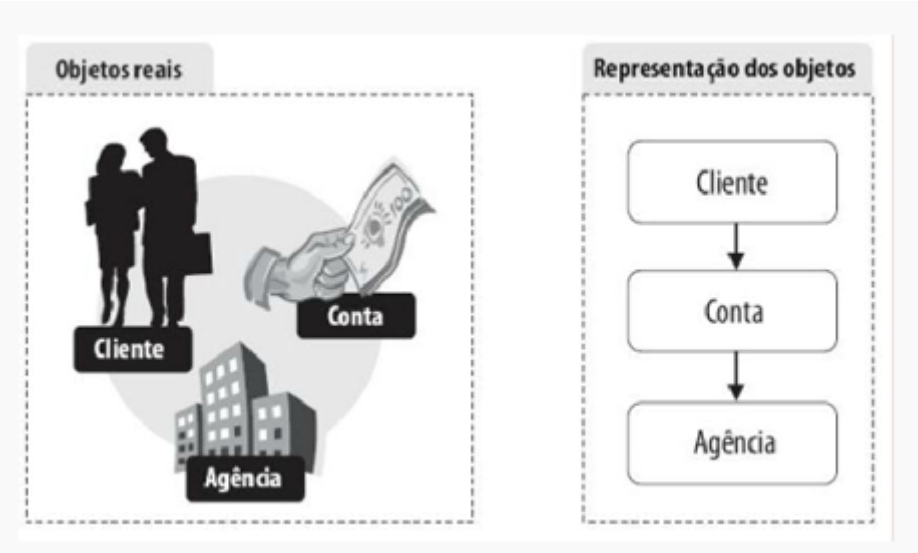
OBJETO = DADOS + FUNÇÕES

Na medida em que são identificados todos os objetos pertinentes a um sistema, já teremos os **dados** e os **procedimentos relacionados.**

Exemplo

Considerando o contexto de um sistema bancário (ilustrado pela imagem a seguir), podemos citar os objetos **agência, cliente e conta**, e a relação "**Cliente possui uma conta em uma agência**".

Ao identificar os objetos, são identificados, também, os dados (atributos) e as funcionalidades (funções) inerentes àquele objeto, conforme ilustra a imagem. O objeto cliente possui os dados nome, endereço, telefone e CPF e, sobre ele, podem ser realizadas, por exemplo, as funções de Incluir Novo Cliente, Excluir Cliente Inativo, Inativar Cliente.



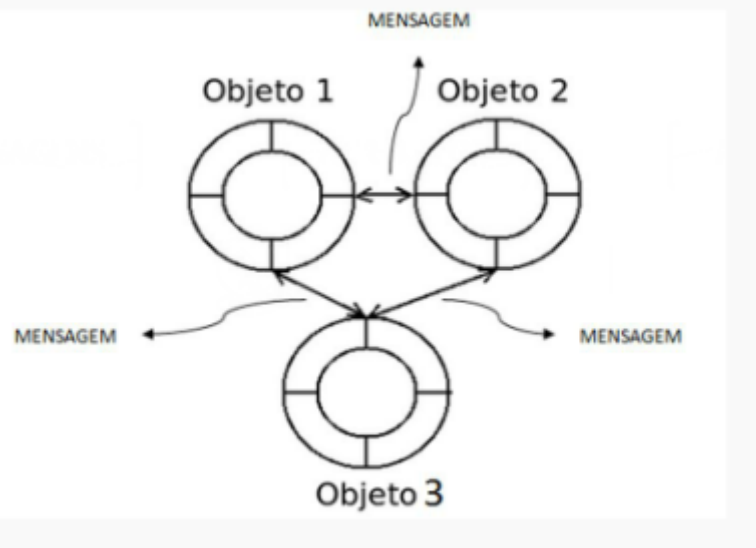
Orientação a Objeto (OO)

Um **modelo OO** tem com entidade fundamental o **Objeto**, que **recebe e envia mensagens, executa procedimentos e possui um estado que, por proteção, apenas ele próprio pode modificar**. Nesse processo, **problemas são resolvidos**, por meio de **objetos**, que **enviam mensagens uns aos outros**.

O **Modelo OO** é formado por alguns **elementos básicos**:

Objeto: É o principal **elemento do modelo OO**. Representam as "**coisas**" a serem **modeladas do mundo real**. Um objeto pode ser algo concreto como um carro, um aluno ou algo abstrato como uma disciplina, um curso. Cada objeto possui os **dados inerentes** a ele, como por exemplo, Nome (José) e Matrícula (201101272201) de um objeto Aluno e possui as **operações que ele executa**, como incluir novo aluno ou alterar dados de um aluno existente.

Mensagens: Quando um objeto deseja que seja executada uma operação de responsabilidade de outro objeto, ele manda uma mensagem a esse objeto informando o que ele deseja que seja feito. A operação desejada será implementada por meio de **um método da classe que recebe a mensagem**, conforme a imagem abaixo:



Atributos: São **dados** que caracterizam o objeto.

Métodos: É um **procedimento** (implementado por uma rotina ou função) que executa uma operação em um objeto, que define parte de seu comportamento.

Classes: É um **conjunto de objetos** com as mesmas características (atributos e métodos). Conforme ilustrado na imagem a seguir, Maria e Pedro são objetos da classe PESSOA, cujas características em comum são:

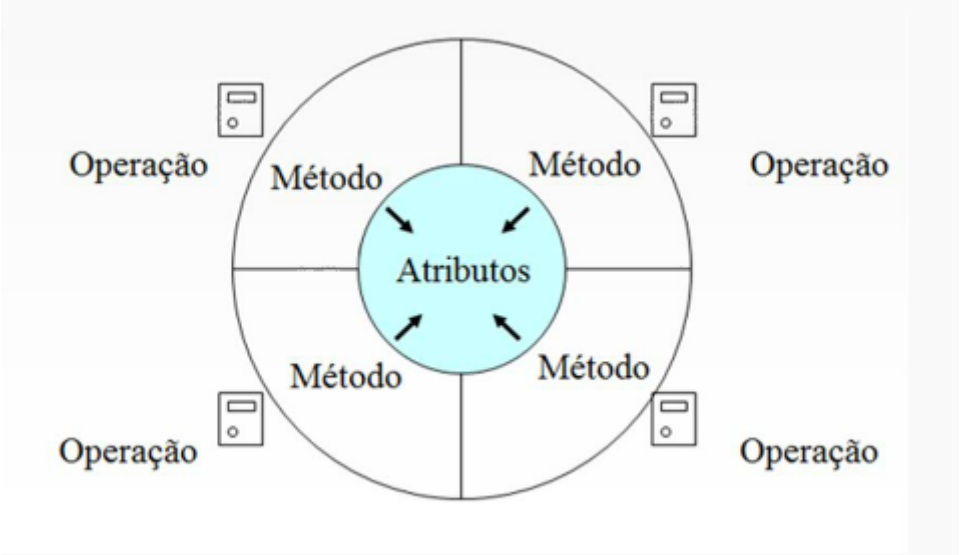
- **Atributos**: Nome, Endereço, Telefone, Idade e Altura.
- **Métodos**: Registrar( ), Matricular( ), Pagar( ), Estudar( ) e Cadastrar( ).

Podemos dizer, portanto, que "Objeto é uma instância de uma classe", isto é, a classe é o molde e o objeto, a "coisa real".

Os pilares do paradigma Orientado a Objetos

A orientação a objetos tem como alicerce alguns princípios fundamentais. Juntos, eles permitem que **classes sejam reaproveitadas, otimizando tempo e custo de desenvolvimento**, além da **segurança no reuso das classes** já usadas e testadas. Observe:

Encapsulamento: Encapsular significa **esconder**, ou seja, o **objeto esconde seus dados (atributos) do acesso indevido de outros objetos**. Os dados somente devem ser acessados por métodos (funcionalidades que implementam o comportamento do objeto) da própria classe, o que pode ser visualizado na figura abaixo:



Herança: **Mecanismo para derivar novas classes a partir da definição de classes** existentes através de um processo de **refinamento**.

Polimorfismo: A palavra **polimorfismo**, deriva do grego, e significa "**muitas formas**". A partir do momento em que uma **classe herda atributos e métodos de uma (herança simples) ou mais (herança múltipla) classes base**, ela tem o poder **de alterar o comportamento** de cada um desses procedimentos (métodos).

Visibilidade: Outro conceito fundamental, é a visibilidade entre as classes. A visibilidade diz respeito ao que **uma classe pode visualizar da outra**. Como princípio, devemos garantir o **encapsulamento**, ou seja, os **atributos devem ser privados** (acessíveis apenas por métodos da própria classe) e **determinados métodos públicos** (acessíveis por todas as classes), e acessar esses dados.

Seguem algumas das conclusões do paradigma OO:

O uso de **técnicas orientadas a objetos facilita o controle da complexidade** uma vez que promove uma melhor **estruturação de seus componentes** e também permite que componentes já usados e validados possam ser reaproveitados.

As hierarquias de classes (herança) são **componentes portáveis entre aplicações, que, se bem projetados, podem ser reutilizados em vários sistemas sem modificação** e, além disso, podem ser estendidos (polimorfismo) sem corromper o que já existe.

Proporciona modularidade trazendo os seguintes benefícios:

- **Reusabilidade**: softwares podem ser escritos com base em componentes já existentes;
- **Extensibilidade**: novos componentes de software podem ser desenvolvidos a partir de outros, já existentes, sem afetar o comportamento do componente de origem.

Um sistema desenvolvido com as características do modelo OO tende a ser **bem estruturado** posto que **os objetos são unidades coesas com interfaces simples** que escondem as suas implementações.

Conceitos de Análise e Projeto Orientado a Objeto

A orientação a objetos enfatiza a identificação, a representação e a organização dos objetos necessários a um sistema.

Antes de adentramos no universo da análise e projeto, sob o enfoque do paradigma Orientado a objeto, vamos tecer rápidos comentários acerca do que sejam as atividades de **Análise e Projeto**, dentro do contexto de **desenvolvimento de software.**

Análise de sistemas significa uma investigação dos problemas e dos requisitos de um contexto, em particular, com vistas ao desenvolvimento de um sistema automatizado. A ideia básica é identificar quais seriam as funções que esse sistema precisa ter, de forma a atender eficientemente as necessidades de seus usuários.

Atividade de análise e projeto

A atividade de análise, por ser muito abrangente, costuma ser dividida em:

- **Análise de requisitos (investigação dos requisitos);**
- **Análise do domínio do problema.**

No contexto específico da **orientação a objetos, análise implica em identificar e descrever os conceitos no domínio do problema**. Já na atividade de análise foca-se na **definição dos objetos de software** e como eles colaboram para que os **requisitos dos usuários** sejam plenamente satisfeitos.

Por outro lado, a **atividade de projeto** denota uma **solução**, voltada a atender aos requisitos, considerando **aspectos de tecnologia**. Em geral, o projeto **enfoca a arquitetura do sistema, definindo suas partes e a relação entre elas**.

Portanto, análise pode ser traduzida em "**faça a coisa certa**", e **projeto** em "**faça certo a coisa**".

Saiba mais

Um sistema desenvolvido sob o paradigma da orientação a objetos representa os objetos que encontramos no mundo real, no problema que estamos tratando, no negócio para o qual o estamos desenvolvendo o sistema. O foco da equipe de desenvolvimento passa a ser **a identificação e modelagem dos objetos** do mundo real que afetam o sistema.

A UML como uma linguagem padrão

A **UML (*Unified Modeling Language)*** é uma l**inguagem padrão para construção de projetos de sistemas, orientados a objeto**, voltada para **visualização, especificação, construção e documentação de artefatos de um sistema**. A UML foi projetada para ser **independente do método de desenvolvimento utilizado**. Ela é uma **linguagem de modelagem,** não é um método de desenvolvimento, nem tão pouco um metodologia ou um processo de desenvolvimento de sistemas, uma vez que não determina a ordem e nem como os diagramas devem ser usados. Simplesmente **disponibiliza os diagramas**, sob as várias visões necessárias ao desenvolvimento de sistemas, e cada empresa utiliza da forma como lhe convém, ou seja, adequando a UML ao seu processo de desenvolvimento de sistema.

A UML é destinada à:

Visualização

A **modelagem gráfica tende a facilitar a compreensão,** permitindo sua i**nterpretação sem ambiguidades**.

Especificação

Permite a **construção de modelos precisos, sem ambiguidades e completos**. A UML atende a esses quesitos sob o ponto de vista da **análise, projeto e implementação**.

Construção

Os diagramas UML podem ser diretamente i**ntegrados a várias linguagens de programação** tais como **Java** e **C++.**

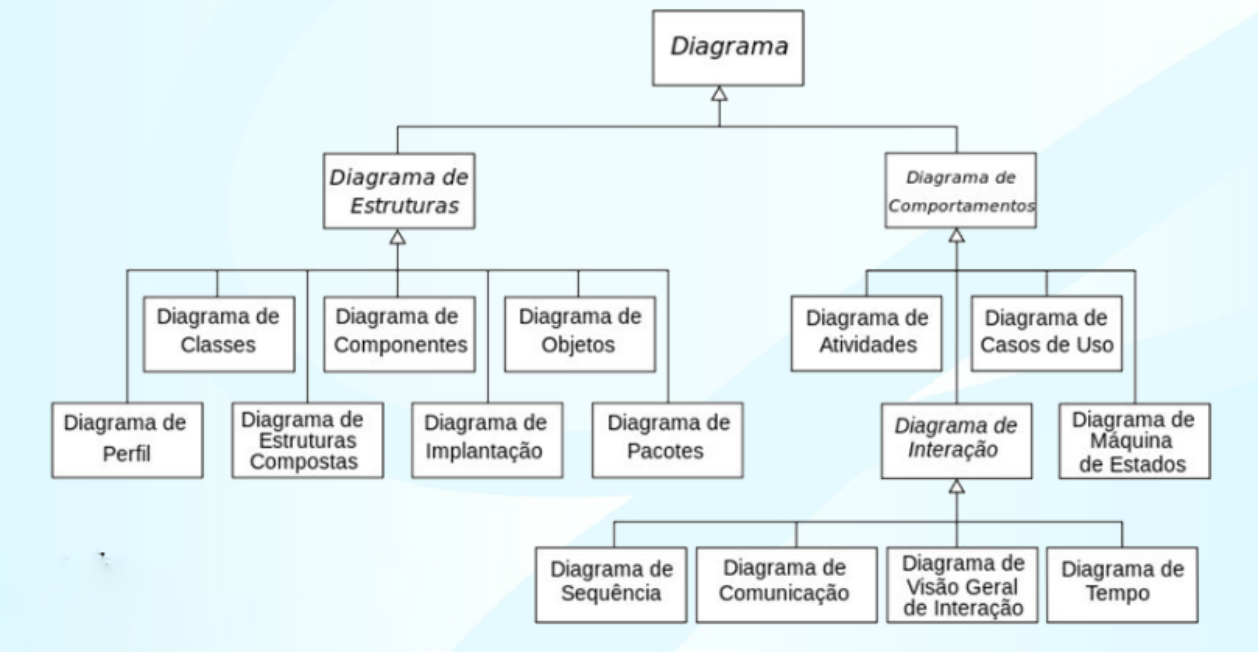
Documentação

A UML abrange a **documentação da arquitetura do sistema** e de todos os seus **detalhes**.

Os diagramas da atual versão da UML

Os **diagramas da UML** são divididos em 2 grupos: **Diagramas de Estruturas e Diagramas Comportamentais**.

No total, eles formam **14 diagramas**, conforme ilustrado pela imagem:



Uso da UML

Como a UML não determina quais diagramas usar nem por onde começar, cada um começa por onde desejar. Nem mesmo os idealizadores da UML conseguem usar todos os diagramas. Geralmente, as empresas escolhem um subconjunto deles e implementam nas fases de seus processos de desenvolvimento de sistemas. O ideal é que você encontre o seu conjunto de diagramas que funcione para o tipo de sistema que desenvolve.

Por exemplo: muitos começam pelo **Diagrama de classes**, na medida em que o consideram o **diagrama mais relevante**. Porém muitos iniciam pelo **Diagrama de Casos de uso**, cuja finalidade é **modelar as principais funcionalidades do sistemas** para atender às necessidades de seus usuários. Ambos estão usando a UML de forma correta e expressando como visualizam a sequencia mais adequada para desenvolver sistemas.

Martin Fowler e **Steve Mellor** propuseram três modos pelos quais pode-se usar a UML no desenvolvimento de sistemas:

- **1- UML como esboço**

  O modo mais usado, onde os desenvolvedores usam a UML como forma de expressar aspectos relevantes de um sistema, esboçando ideias e alternativas do que pretende fazer. É uma possibilidade de discussão com a equipe de desenvolvedores. Seu foco é a comunicação seletiva em vez de especificação completa. Geralmente, são usados desenhos informais, sem ferramentas e cujo objetivo é a discussão de ideias visando à construção de software de forma colaborativa. O modo UML, como esboço, está mais compatível com as metodologias ágeis, que preconizam mais código e menos documentação.

2- UML como projeto

Foca a completeza. Aqui a ideia é construir um projeto completo para ser codificado por programadores, valendo-se de ferramentas case para melhor entendimento dos modelos pela equipe. O modo UML, como projeto, está mais alinhado com os processos de desenvolvimento mais complexos, como processos iterativos e incrementais, como o PU (processo unificado) implementado através da ferramenta RUP (Rational Unifiec Process) prototipação.

- **3- UML como linguagem de programação**

Onde os desenvolvedores desenham os diagramas que são compilados para o código executável e a UML se torna o código fonte. Exige ferramentas mais sofisticadas. O modo UML, como linguagem de programação, tende a ser mais usado em modelos de desenvolvimento de prototipação.

Processos Iterativos de desenvolvimento e a UML

Um processo de desenvolvimento de software descreve uma **abordagem** para organizar as atividades relacionadas com a **construção**, a **implantação** e a **manutenção de sistemas**.

Os processos mais populares hoje são os **iterativos**, como:

- O **PU** (Processo Unificado), em particular o **RUP** (Processo Unificado da Rational);
- As metodologias ágeis, como o **XP** (*eXtreming Programming*) e **Scrum**.

Mas o que são os processos iterativos?

São processos onde o **ciclo de vida do sistema** é dividido em uma **série de mini projetos**, curtos, preferencialmente de **duração fixa** (por exemplo, 3 meses), denominados **iterações**.

A UML e os resultados em cada fase

Usar a UML, em um processo de desenvolvimento de sistemas, não implica, necessariamente, na elaboração de documentos ou uso de uma **ferramenta case.**

Muitas equipes, por exemplo, usam UML para desenhos à mão em reuniões e para discussão de ideias.

Uma **ferramenta case** é um p**rograma que auxilia aos membros da equipe de desenvolvimento no estudo, modelagem e construção do sistema**, possibilitando que vários diagramas possam ser elaborados em conjunto, representando a estrutura e comportamento do sistema a ser desenvolvido.

Saiba mais

Para saber mais sobre metodologias de desenvolvimento ágil, acesse os seguintes sites:
DesenvolvimentoAgil. Disponível aqui: http://www.desenvolvimentoagil.com.br/xp/
Scrum. Disponível aqui: https://www.scrum.org/

Leia os seguintes textos:

SCRUM. Disponível aqui: http://www.desenvolvimentoagil.com.br/scrum/

Getting Started With UML. Disponível aqui: http://www.uml.org/

Manifesto ágil. Disponível aqui: http://www.desenvolvimentoagil.com.br/xp/manifesto_agil

Introdução ao Processo Unificado. Disponível aqui: http://www.devmedia.com.br/introducao-ao-processo-unificado/3931

O que vem na próxima aula

- Levantamento e modelagem de Requisitos;
- Modelos de casos de uso, apresentando os principais conceitos e elementos do Diagrama de Casos de Uso, conforme preconizado pela UML, e as funcionalidades que atenderão aos requisitos do sistema.

CONCLUSÃO

Nesta aula, você:

- Reconheceu o conceito de modelagem de sistemas e a relevância dos diagramas para representação da realidade;
- Identificou o paradigma Orientado a Objeto, apresentando suas caraterísticas, princípios e principais pilares de sustentação;
- Distinguiu os principais conceitos de análise e projetos Orientado a objetos;
- Definiu a UML, sua origem, objetivos e principais diagramas;
- Reconheceu os principais conceitos dos processos iterativos;
- Estabeleceu como alinhar a UML a um processo de desenvolvimento de software, especialmente, um modelo iterativo.

Referências

BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. **UML** — Guia do Usuário. 2. ed. Rio de Janeiro: Elsevier, 2005. cap. 1 e 2.

FOWLER, Martin. **UML essencial** — um breve guia para a linguagem padrão. 3. ed. Porto Alegre: Artmed, 2005. cap. 1

LARMAN, Craig. **Utilizando UML e padrões?** uma introdução à análise e ao projeto orientados a objetos e ao processo unificado. 3. ed. Porto Alegre: Artmed, 2007. cap. 2.