MÚLTIPLAS PERSPECTIVAS DE OBJETOS NO CONTEXTO EDUCACIONAL

(Minicurso do SBIE 2000 – Simpósio Brasileiro de Informática na Educação – 8 a 10 de novembro de 2000)

André Santanchè
UNIFACS - Universidade Salvador
santanche@unifacs.br
www.geocities.com/santanche

Cesar Augusto Camillo Teixeira
UNIFACS - Universidade Salvador
cesar@unifacs.br

1. Apresentação

No final da década de 60 surge uma poderosa metodologia de programação de computadores, que se destacou por adotar a metáfora de objetos intercomunicantes para representar os componentes de um programa. A Programação Orientada a Objetos foi o ponto de partida para um paradigma que se expandiu para outras áreas da computação, tal como a análise e projeto de sistemas, bancos de dados, etc. Nos últimos anos modelos computacionais baseados em objetos passaram a ser utilizados em outros contextos, inclusive no da educação.

Este texto apresenta um estudo dos fundamentos da tecnologia de objetos e suas múltiplas aplicações na educação. Em especial, são analisadas as interfaces do modelo de objetos com diversos outros modelos utilizados em atividades de ensino-aprendizagem, envolvendo tecnologia, tais como componentes, agentes e autômatos celulares.

O texto inicia com uma introdução que procura responder a questão “Por que utilizar objetos?”. No tópico seguinte é apresentada uma síntese das origens do paradigma e o que motivou seu surgimento.

Em “Tecnologia de Objetos” são resumidos alguns dos conceitos mais significativos, que surgiram com a programação orientada a objetos e vem se reconfigurando em cada novo modelo. No tópico “Relações e Influências” são analisados os conceitos de Componentes, Agentes e Autômatos Celulares e sua relação com os objetos.

Estes quatro tópicos iniciais introduzem os fundamentos para o que é apresentado em seguida, que consiste em uma panorâmica do uso de tecnologias em atividades educacionais nas quais os objetos estão envolvidos.

Entre os sistemas explorados estão alguns voltados à construção de modelos animados, adequados para a simulação de fenômenos, tal como o LiveWorld, AgentSheets e Homos. Estes sistemas utilizam metáforas bastante próximas à realidade do aprendiz de modo que ele possa modelar e estudar fenômenos que de outra maneira exigiriam ferramentas bem mais complexas.

Por outro lado, existem sistemas que se apresentam como caixas de ferramentas, dotados de componentes com os quais o aprendiz poderá construir uma grande diversidade de outros modelos para estudo. Nesta categoria estão o: E-Slate, Casa Mágica e SimCalc.

Muitos destes sistemas tem recentemente sido combinados no projeto ESCOT que, baseado em componentes de software, estabelece um modelo de integração entre eles segundo sua especialidade.

Também são apresentados esforços para se estabelecer um padrão de metadados que descreva Objetos de Conhecimento e sua aplicação, em que estão envolvidos o IEEE com seu padrão denominado LOM - *Learning Objects Metadata* associado ao IMS *Global Learning Consortium*, que estabeleceu uma representação deste padrão na linguagem XML.

Em síntese, este trabalho se propõe a apresentar, de forma introdutória, os principais conceitos que envolvem objetos e sua aplicação em atividades de ensino-aprendizagem, apresentando alguns exemplos de trabalhos e programas que seguem este paradigma.

2. Introdução

Uma análise sobre as atuais interfaces computacionais conduz a uma reflexão sobre os motivos que as tem tornado mais simples e intuitivas no seu uso.

Após a vitória que consistiu na concepção e concretização dos primeiros computadores, os seus criadores se debruçaram numa segunda tarefa não menos trabalhosa ou importante: elaborar mecanismos que simplificassem a interação entre o computador e o homem.

2.1. Interfaces, Linguagens e Computadores

De fato, os primeiros computadores eram máquinas desajeitadas, cuja programação e operação era feita através de conexões físicas de cabos e exigia um profundo conhecimento da arquitetura dos mesmos.

As primeiras linguagens de programação e interfaces com sistemas operacionais tomaram como base para seu modelo de concepção a própria arquitetura do computador, ou seja, o centro era o computador. Por este motivo, os sistemas operacionais reuniram comandos para controle dos dispositivos, organização da memória, execução de tarefas, etc. As linguagens de programação, por sua vez, representaram de forma simplificada e produtiva seqüências de instruções dadas ao computador.

Em ambos os casos, o modo de se pensar, organizar e representar a solução de problemas é necessariamente condicionado à forma como o computador organiza e representa os dados.

> *"Mas programação, que é o único nível de uso em que todo o poder da computação como uma ferramenta intelectual pode ser realizado, ainda está centralizado em torno dos modelos tradicionais de computação. Neste nível, um computador é visto como um dispositivo para desempenhar cálculos matemáticos, ou executar uma seqüência de operações de uma*

> *rotina – não como um mundo dinâmico em que atividades vivas podem acontecer.”* [TRA96]

Cada instrução escrita em uma linguagem computacional ou comando dado a um sistema operacional, é passível de um mapeamento para uma estrutura interna de representação do computador, por este motivo o usuário, ao interagir com o computador através destes meios, está lidando com uma metáfora que lhe é apresentada com o objetivo de simplificar a comunicação ou aumentar sua produtividade.

As metáforas escolhidas nas primeiras linguagens e interfaces, partindo do computador e sua estrutura, são adequadas quando o indivíduo que interage com a máquina é um profissional da área de informática. Por outro lado, quando se trata de usuários leigos isto se torna um entrave.

2.2. Interface e Novas Metáforas

Uma metáfora, segundo o dicionário Aurélio, é um “Tropo em que a significação natural duma palavra é substituída por outra com que tem relação de semelhança” [FER89].

Lakoff e Johnson ampliam a interpretação do termo metáfora, para além dos limites da linguagem, analisando sua influência no pensamento e ação do indivíduo.

> “*Metáfora é para muita gente um dispositivo da imaginação poética e da alegoria retórica – um assunto de linguagem extraordinária ao invés de ordinária. Além disto, metáfora é tipicamente vista como característica só da linguagem, um corpo de palavras ao invés de pensamento e ação. Por esta razão, muitas pessoas pensam que elas podem avançar perfeitamente bem sem metáforas. Nós percebemos, ao contrário, que essa metáfora impregna cada dia da vida, não somente na linguagem mas no pensamento e ação. Nosso sistema conceitual normal, em termos do que nós pensamos e agimos, é fundamentalmente metafórico por natureza*”. [LAK80]

A compreensão da metáfora, inerente às interfaces e linguagens computacionais, passou a ser usada como um instrumento para a sua melhoria, de modo que os ambientes apresentados ao usuário lhe sejam familiares e se encontrem dentro de seu domínio de conhecimento. Cabe ao computador e um conjunto de programas apropriados realizar um “mapeamento metafórico”, ou seja, converter as ações do usuário pautadas neste novo modelo, em comandos que estão de acordo com o funcionamento específico da máquina.

Os atuais Sistemas Operacionais são um exemplo típico. Com o surgimento da interface gráfica, criou-se a metáfora da mesa de trabalho, onde o computador cria um ambiente virtual para o usuário. Os programas são apresentados em janelas que podem ser deslocadas e reorganizadas pela tela e o acionamento de um programa pode ser feito através de um ícone ou um menu de opções.

Desta maneira, estes sistemas tornaram-se mais fáceis de serem manuseados, mais acessíveis e amigáveis. Isto se deu graças a um deslocamento da perspectiva. A arquitetura

e funcionamento do computador deixaram de ser o centro a partir do qual são elaborados os modelos de interface. O centro tornou-se então, o próprio indivíduo, sua maneira de observar e conceber o mundo que o cerca.

Uma modificação semelhante tem se dado com as modernas linguagens de programação.

> *"As linguagens usadas para programação são definidas, habilitadas e limitadas por seus modelos e metáforas subjacentes. Enquanto a ciência da computação aspira em direção a definições formais sobre seu tema, a tarefa prática de entender entidades computacionais depende da técnica informal de tomar emprestado a terminologia e a estrutura de domínios familiares através de mapeamentos metafóricos"*. [TRA96]

É neste contexto que emerge a Metáfora dos Objetos.

3. Um pouco de história

Objetos são uma expressão cada vez mais natural para os envolvidos com a computação. O conjunto de princípios que deram origem à Orientação a Objetos surgiu no final da década de 60, através da SIMULA [HOL92], uma linguagem inicialmente projetada para realizar simulações por eventos discretos que posteriormente foi modificada para uma versão genérica. A versão do SIMULA lançada em 1967 (SIMULA 67) apesar de nunca ter sido muito difundida, foi a primeira linguagem implementar princípios da Programação Orientada a Objetos.

No início da década de 70 Alan Kay, pertencente ao projeto Dynabook, liderou os esforços para a concepção de uma linguagem de programação de computadores cujos princípios estivessem em sintonia com o projeto.

> *"Este 'Dynabook' foi baseado na visão de computadores pessoais baratos do tamanho de um caderno, tanto para adultos quanto crianças, com a capacidade de lidar com todas as suas respectivas necessidades de informação"*. [KRE98]

Emerge deste contexto a linguagem Smalltalk. Seu projeto tencionava tornar o processo de construção de aplicações acessível a quaisquer usuários, sem que eles tivessem o domínio de linguagens de programação voltadas para iniciados na computação.

Daniel Ingals resume este propósito enunciando o princípio mestre do projeto que, como ele mesmo afirma, é mais social do que técnico: "*Se um sistema serve ao espírito criativo, ele tem que ser inteiramente compreensível para um único indivíduo*". [ING81]

Apesar de seu propósito claramente voltado a permitir que usuários não especialistas em computação produzissem suas próprias aplicações, o Smalltalk, durante muitos anos após o seu lançamento serviu como base para uma nova configuração das linguagens de programação, que passaram a trabalhar com objetos.

A Programação Orientada a Objetos trouxe, deste modo, grandes benefícios à programação de computadores. No entanto, de um modo geral continuou no domínio dos especialistas em computação.

4. Tecnologia de Objetos

A Orientação a Objetos avançou as fronteiras da programação e se mostrou mais do que uma técnica, introduzida pelo SIMULA e Smalltalk e seguida por outras linguagens. Seu conjunto de princípios e modelo de organização de idéias se mostrou igualmente aplicável em outras áreas da computação, como a análise e projeto de sistemas, modelagem de bancos de dados, etc.

Os objetos são, neste contexto, uma metáfora bem mais próxima da maneira como percebemos o mundo que aquelas cuja origem se fundamentam no funcionamento do computador.

> *"A programação Orientada a objetos (POO) é um exemplo interessante de uma metodologia de programação explicitamente organizada em torno de uma poderosa metáfora. Em POO, objetos computacionais são representados metaforicamente em termos de objetos físicos e objetos sociais. Como objetos físicos, eles podem ter propriedades e estado, e como objetos sociais, eles podem se comunicar e responder a comunicações"*. [TRA96]

Apesar dos variados contextos, as diversas linguagens, metodologias, padrões e modelos "orientados a objetos" obedecem, em graus variáveis, a um conjunto de princípios que configuram um paradigma.

Segundo Thomas Kuhn, paradigma "*[...] indica toda a constelação de crenças, valores, técnicas, etc..., partilhadas pelos membros de uma comunidade determinada [...]"* [KUH87]. Podemos inicialmente estabelecer alguns princípios que norteiam o paradigma dos objetos. Estes princípios foram originalmente aplicados nas linguagens de programação e em seguida transpostos para outros domínios da informática.

4.1. Princípios do Paradigma

A seguir estão sintetizados alguns dos princípios que compõe a constelação do Paradigma de Orientação a Objetos. Foram selecionados aqueles princípios que estão diretamente relacionados com as análises, que serão realizadas a seguir, referentes ao uso de objetos educacionais. Por este motivo, alguns conceitos fundamentais do paradigma, porém menos importantes para a nossa análise, foram omitidos.

Objeto

"O que se apresenta à percepção com um caráter fixo e estável". [FER89]

Seu conceito corresponde intuitivamente à percepção que temos de elementos individuais no universo que nos cerca, convenientemente isolados sobre a forma de objetos.

> *"A mente observa um vasto universo da experiência, simultaneamente imediato e gravado. Alguém pode derivar um senso de unidade com o universo simplesmente deixando esta experiência ser, da mesma forma que ela é. Contudo, se alguém deseja participar, literalmente tomar parte, no universo, ele precisa extrair distinções. Agindo assim, este identifica um objeto no universo, e simultaneamente, todo o restante se torna não-aquele-objeto"*. [ING81]

Para entendermos como os sistemas baseados em objetos caracterizam seus objetos, tomemos um objeto desconhecido para alguns dos leitores, um animal conhecido como "ornitorrinco" (aqueles que o conhecem assumam, por favor, o contrário).

Tentaremos fazer uma descrição de um ornitorrinco de modo que o leitor possa obter alguma noção referente ao animal:

Um ornitorrinco é um animal aquático dotado de pêlos, que possui um bico igual ao de um pato e quatro patas usadas como nadadeiras. Apesar de colocar ovos como os pássaros o ornitorrinco amamenta a sua prole, tal como os mamíferos (para quem não conhece o ornitorrinco é difícil acreditar que exista).

Para realizar esta descrição foram apresentadas as características do ornitorrinco:

- dotado de pêlos;
- bico igual ao de um pato;
- patas usadas como nadadeiras.

Mas as características não são suficientes para entender o que é um ornitorrinco, foi necessário também se descrever seu comportamento:

- vida aquática;
- colocar ovos como os pássaros;
- amamentar sua prole.

Por este motivo, os objetos, nos sistemas baseados em objetos, são caracterizados por três aspectos: identidade, características (estado interno) e comportamento.

Além das características e do comportamento é necessária a identidade para se caracterizar um objeto. Quando falamos sobre o ornitorrinco, não estamos fazendo referência a um ornitorrinco específico, mas a todos que pertencem à espécie. A identidade individualiza

um ornitorrinco específico, vivente em algum lugar da Austrália, único e diverso de todos os demais.

Diferentes objetos podem ter uma característica em comum que, no entanto, se apresenta de forma diversificada em cada um deles. Por exemplo, todos os ornitorrincos tem uma cor de pêlo, no entanto, indivíduos diferentes podem ter tonalidades diversas desta cor. Isto caracteriza um certo estado da característica cor.

As características e seus respectivos estados são representados em modelos computacionais de objetos por variáveis e valores que determinam seu estado. Por exemplo, a característica "altura" pode ser representada por uma variável cujo valor (número real) representa a altura (estado) de um objeto. A estas variáveis chamamos de **atributos**.

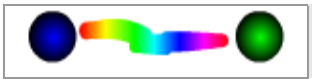
Uma esfera pode ter suas características descritas da seguinte maneira:

peso: 200 g
tamanho (raio): 60 cm
elasticidade: alta
cor: vermelha

Cada objeto é capaz de realizar um conjunto de atividades que determinam o seu comportamento. Por exemplo, um objeto representando uma bola pode ter o comportamento de saltar ao receber um estímulo. Este comportamento, por sua vez, sofre influência do estado interno inerente ao objeto. Assim, uma bola cuja elasticidade (estado) seja elevada, tenderá a pular mais alto quando se chocar com o chão do que uma outra bola nas mesmas condições porém menos elástica.

Podemos concluir que o mesmo comportamento pode variar em objetos equivalentes, dependendo do estado interno dos mesmos.

Em modelos computacionais o comportamento de um objeto é representado através de módulos que contém algoritmos computacionais denominados **métodos**. O procedimento do método pode variar de acordo com os valores dos atributos.



Mensagem

A comunicação entre os objetos é realizada através de mensagens. É através delas que um objeto solicita a outro que realize determinada tarefa.

Em sistemas computacionais baseados em objetos, a cada mensagem recebida é disparado um método do objeto que ativa uma ação.

Uma aplicação baseada em objetos se configura como uma rede (*framework*) de objetos que se intercomunicam através de mensagens. Isto pode ser observado no exemplo apresentado na figura 4.1.1:

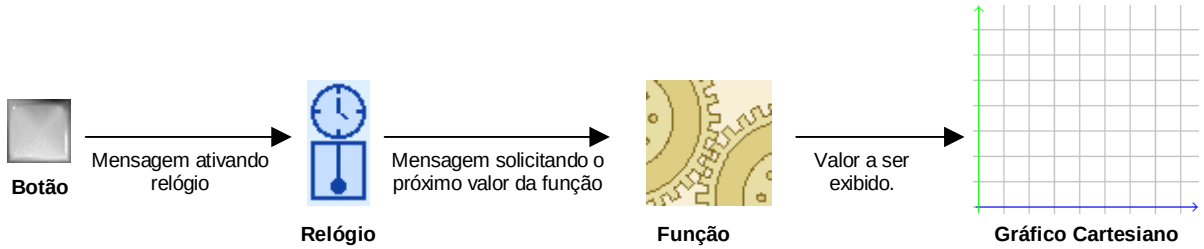
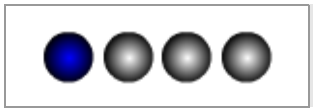


Figura 4.1.1: Aplicação em elaborada em Casa Mágica que apresenta o gráfico de uma função de segundo grau a partir de uma framework de objetos

O modelo da figura 1 é composto de quatro componentes. O botão é responsável por iniciar o processo enviando uma mensagem de ativação para o componente relógio. O relógio inicia então uma seqüência de batimentos que irão gerar mensagens enviadas para o componente função. A cada mensagem recebida o componente função calcula um par de valores e os envia para o componente gráfico cartesiano, que exibe os valores recebidos.



Classe

"Numa série ou num conjunto, grupo ou divisão que apresenta características ou atributos semelhantes". [FER89]

Muitos objetos possuem comportamento semelhante, ou seja, respondem às mesmas mensagens do mesmo modo. Além disso, apresentam estruturas internas equivalentes, caracterizadas pelos seus atributos. A percepção destas semelhanças conduziu a pesquisadores como Carl Linné a produzir uma taxonomia para a classificação dos seres vivos.

Figura 4.1.2: Algumas classes da taxonomia de Carl Linné

Para elaborar esta taxonomia Carl Linné reuniu os animais segundo as características e comportamentos comuns e estabeleceu vários níveis de classificação. As classes, como as apresentadas na figura 4.1.2, são uma entidade abstrata que representam um conjunto de animais.

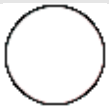
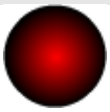
Modelos orientados a objetos tem a capacidade de agrupar (classificar) seus objetos sob a forma de classes. Inversamente, os objetos passam a ser instâncias de uma classe.

Bertrand Meyer considera a classe o conceito central da tecnologia de objetos, pois ela é a "noção básica, de onde tudo, na tecnologia dos objetos, deriva" [MEY97]. De fato, ao contrário de como acontece na vida real, onde inicialmente conhecemos os objetos para depois classificá-los, quando elaboramos modelos baseados em objetos, inicialmente concebemos a classe e a partir dela instanciamos os objetos.

Quando realizamos uma classificação de objetos, identificamos o seu comportamento e as características que eles possuem em comum. Por este motivo, as classes definem:

- os **atributos** que irão descrever o objeto;
- os **métodos** que definem o comportamento dos mesmos.

Uma classe define apenas que atributos irão descrever o objeto. Cada objeto estabelece valores específicos (estado) para esses atributos:

Classe	Objeto	Objeto	Objeto
 **peso** **raio** **cor**	 **peso:** 200 g **raio:** 60 cm **cor:** vermelha	 **peso:** 200 g **raio:** 60 cm **cor:** azul	 **peso:** 50 g **raio:** 30 cm **cor:** verde

Os métodos da classe são os mesmos para todos os objetos, porém, como já foi abordado, podem sofrer variações no comportamento de acordo com o estado interno do objeto.

Encapsulamento

> *"Na programação orientada a objetos, os objetos comunicam-se entre si através de mensagens. A única coisa que um objeto conhece sobre outro objeto é a sua interface de comunicação. Os dados e a lógica de cada objeto são mantidos escondidos dos outros objetos. Em outras palavras, a interface encapsula o código e os dados do objeto".* [IBM]

O princípio do encapsulamento estabelece uma divisão entre:

Interface: partes do objeto que se relacionam com o exterior;
Implementação: dados e código que implementam o comportamento do objeto.

O encapsulamento é fundamentado na capacidade que as classes tem de ocultar informações. Deste modo, "é possível para o autor da classe especificar que uma característica está disponível para todos os clientes, para nenhum cliente, ou para clientes específicos" [MEY97]. Isto é bastante útil para a construção de interfaces limpas e objetivas, que podem ser variadas conforme o domínio.

Além de controlar o que o usuário do objeto pode ver ou utilizar, esta separação entre a interface e a implementação tem sido um mecanismo utilizado para comunicar objetos de universos variados.

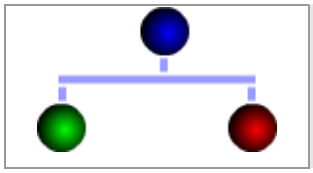
A perspectiva, introduzida pelas redes de computadores, de intercâmbio e integração dinâmica dos dados de máquinas distintas incentivou a construção de objetos que se intercomunicam e trabalham cooperativamente através da rede.

Os objetos são ideais para este tipo de atividade, como destaca Frank Manola:

> *"Há uma crescente concordância que modelar um sistema distribuído como uma coleção distribuída de objetos interativos provê uma framework apropriada para ser usada na integração de recursos computacionais heterogêneos, autônomos e distribuídos (HAD)".* [MAN98]

A separação entre a interface e a implementação, promovida pelo encapsulamento, permite isolar as particularidades de sistemas diversos dentro da implementação de objetos, conduzindo a uma comunicação entre estes somente através de mensagens. Estas, por sua vez, dependem exclusivamente das interfaces, que podem ser compatíveis mesmo em sistemas muito diferentes.

A ampla rede que forma a Internet, dotada de uma grande heterogeneidade de sistemas, tem se mostrado o domínio mais promissor para a difusão desta abordagem.



Herança

Estudando em mais detalhes a taxonomia de Carl Linné, encontramos diversos níveis de classificação. Níveis mais especializados se reúnem em um mais genérico. Por exemplo, o sub-filo Vertebrata reúne um conjunto de classes (figura 4.1.3) que possui características e comportamentos comuns: todos os animais têm uma coluna vertebral na fase adulta e tem o cérebro protegido por uma caixa craniana.

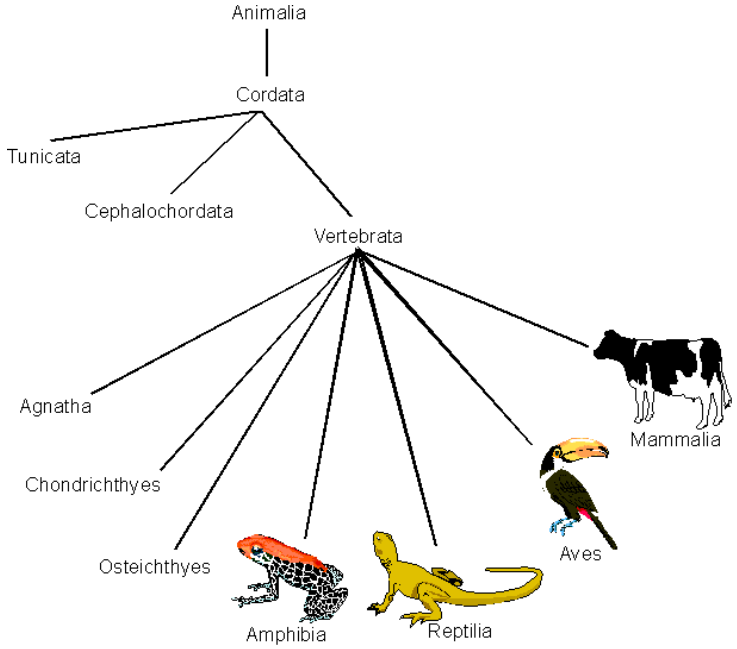


Figura 4.1.3: Algumas classes da taxonomia de Carl Linné agrupadas em outros níveis de classificação

O sub-filo Vertebrata, por sua vez, em conjunto com outros dois sub-filos, fazem parte de uma classificação mais genérica denominada filo Cordata. Cada animal de um nível de classificação mais especializado possui todas as características e comportamento do nível mais genérico a que ele faz parte, acrescentando particularidades próprias. Por exemplo, animais da classe Mammalia possuem todas as características e comportamento do sub-filo Vertebrata mais algumas específicas como o hábito de amamentar sua prole.

Este tipo de procedimento também é utilizado na tecnologia de objetos e é chamado de herança. A herança é o mecanismo utilizado para se definir uma nova classe de objetos a partir de uma “imitação, refinamento ou combinação” [MEY97] de outras classes já existentes. Isto se torna possível tomando uma classe como herdeira de outra.

Uma classe herdeira de outra, denominada subclasse, dispõe dos mesmos atributos definidos na classe superior, denominada superclasse, e provê acesso também aos métodos da superclasse. Além dos atributos herdados, a subclasse pode acrescentar atributos e métodos próprios (exclusivos).

4.2. Modelo de Objetos

Como já foi abordado anteriormente, o paradigma de orientação a objetos se expandiu para diversos domínios e assumiu diversas configurações. Muitos padrões, metodologias, modelos e linguagens adotaram parcial ou integralmente seus princípios.

Torna-se então um desafio compatibilizar estas configurações e encontrar os pontos de intersecção entre elas.

Este é o desafio do Comitê H7 (*Object Information Management*) que faz parte do *National Committee for Information Technology Standards* (NCITS) [MAN97]. Este Comitê se empenha nas seguintes tarefas:

- "Baseado nas recomendações do *Object-Oriented Database Task Group* (OODBTG), desenvolver um relatório técnico para posteriormente refinar o modelo referencial de gerenciamento de informações baseado em objetos".
- "Trabalhar ativamente com outras organizações para adquirir consenso e harmonia em áreas potenciais para padronização de objetos". [MAN97]

O Comitê adotou o termo *Object Model* (Modelo de Objetos) para identificar os modelos subjacentes associados a atuais propostas baseadas em objetos e produziu uma matriz (*Object Model Features Matrix*) onde são apresentadas diversas linguagens e especificações e são confrontados seus modelos de objetos.

Frank Manola, participante do Comitê, também adota o termo *Object Model* em um estudo referente a modelos de objetos adequados para *Web* (*Web Object Model*) [MAN99]. Ele destaca três aspectos fundamentais para qualquer modelo de objetos:

- Estruturas de dados que representam o *estado do objeto*.
- Meios para associar comportamento (métodos do objeto) ao estado do objeto.
- Meios para os métodos do objeto acessarem e operarem sob seu estado.

5. Relações e Influências

As diversas perspectivas de uso da tecnologia na educação adotam metáforas que mediam a interação entre o aprendiz e a máquina. Como foi apresentado no item "2.1. Inteface, Metáfora e Linguagem", estas metáforas cumprem simultaneamente a função de modelo mental e modelo computacional, sobre o qual o software é estruturado e montado.

A seguir apresentamos alguns modelos computacionais e suas metáforas subjacentes, que tem tido grande influência no uso das tecnologias na educação.

5.1. Componentes

A metáfora dos componentes de software, tal como a dos objetos, é proveniente do domínio da programação de computadores. Apesar da grande semelhança entre os componentes e os objetos, existem alguns aspectos que os distinguem.

Componentes de software têm a capacidade de encapsular dentro de um módulo uma coleção de programas interligados entre si, formando um elemento autônomo. Esta é a principal característica dos componentes, que os torna bastante semelhantes a objetos. Porém, os componentes não implementam obrigatoriamente alguns conceitos típicos do paradigma de objetos, tal como a herança. Isto pode ser notado em linguagens dotadas de componentes de software sem, no entanto, serem orientadas a objetos.

Apesar de não serem obrigatórios os princípios do paradigma de orientação a objetos são desejáveis e podem ser associados aos componentes sem restrições. Por este motivo, cada vez mais as linguagens de programação os tem configurado como um subconjunto específico de objetos.

Os componentes são projetados como pequenas peças, facilmente interligáveis para a construção de um modelo mais complexo. Podem ser comparados a pequenas peças de Lego® que são projetadas para serem combinadas na composição de algo maior.

Cada componente é facilmente configurado para ser utilizado em uma aplicação específica e pode ser interligado com outros, sem que o usuário saiba como ele foi implementado. Jeremy Roschelle ao se referir a componentes de software faz uma analogia:

> "*Somente poucas pessoas podem construir seu próprio sistema estéreo interligando transistores e outros componentes no nível do circuito, mas muitas pessoas podem montar um sistema estéreo adequado a suas necessidades, ligando cabos ao seu amplificador, receiver, disc player, prato e alto-falantes preferidos*" [ROS98]

Os componentes foram os responsáveis pelo advento das ferramentas visuais que os utilizam como blocos para a construção de aplicações através da manipulação direta. Para isto elas dispõem de bibliotecas de componentes visuais.

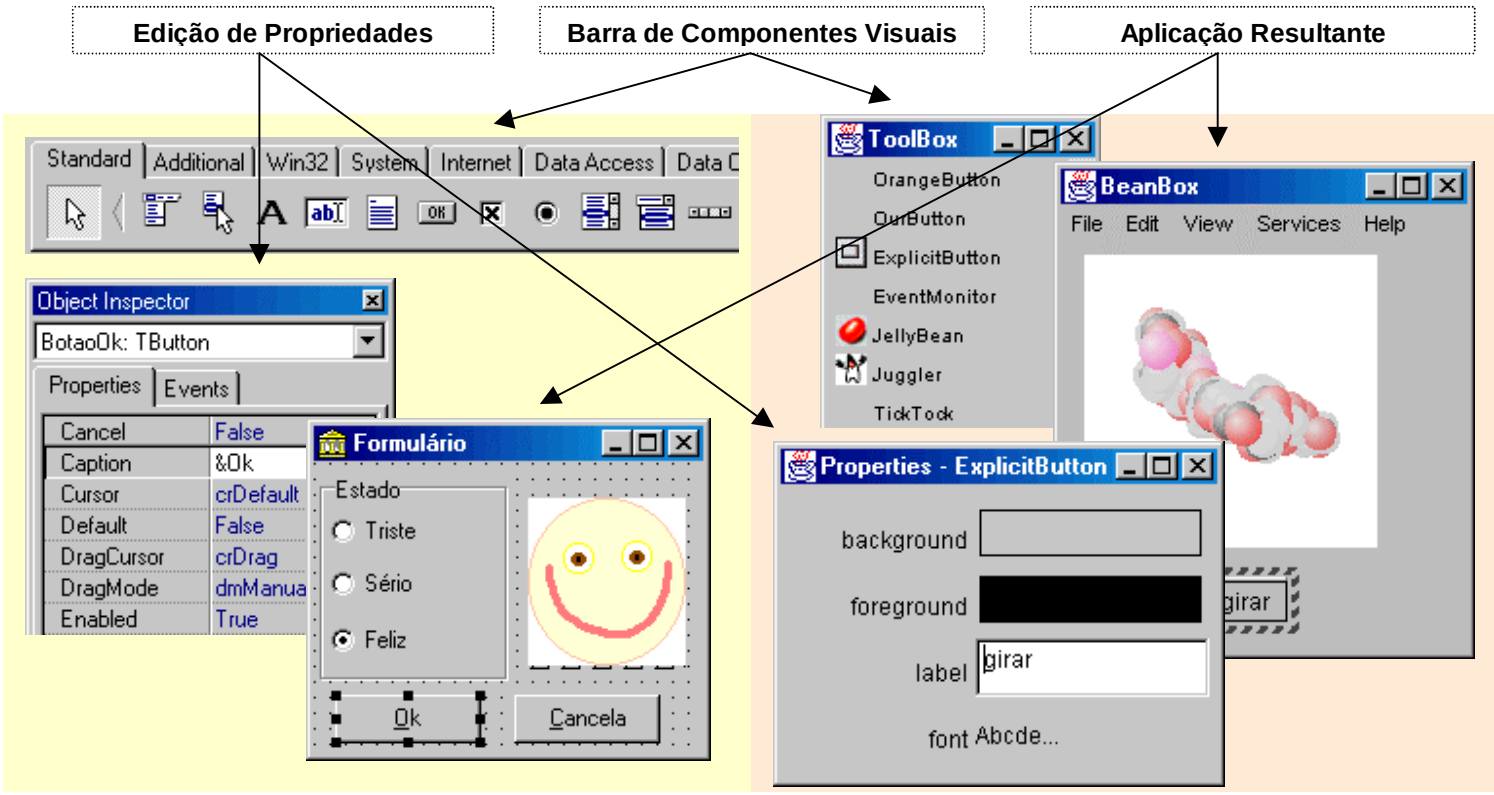


Figura 5.1.1: Ambientes visuais para a elaboração de aplicações baseadas em componentes em Delphi e em Java (beans) respectivamente.

Como pode ser observado na Figura 5.1.1, as ferramentas visuais baseadas em componentes guardam muitas semelhanças na forma como constroem e modificam suas aplicações.

A partir de uma barra de seleção, o usuário pode escolher um componente e coloca-lo na janela da aplicação. Esse componente pode ser configurado, seja através de sua manipulação direta na área da aplicação ou pela modificação dos valores de suas propriedades.

5.2. Agentes

O conceito de "agente" pode ter grandes variações a depender do domínio em que esteja sendo empregado. Por este motivo, nossa tarefa inicial é definir, em linhas gerais, algumas das interpretações mais usadas e destacar a que iremos nos referir doravante.

Michael Travers ao escrever "*Programming with Agents*" [TRA96] se reporta ao termo agente usado por Marvin Minsky em seu livro "*Society of Mind*" [MIN87][1]. Minsky descreve a mente como uma sociedade de pequenos agentes:

[1] Em especial no tópico "1.3 *Influences and Inspirations*" em seu livro "*Programming with Agents*", Michael Travers faz uma explanação mais aprofundada do uso do conceito de agentes em sua tese e o diferencia do uso deste mesmo termo em outros domínios.

> "*Cada agente mental, individualmente, só pode realizar tarefas simples, que não exijam, de modo algum, qualquer mente ou pensamento. Contudo, ao agruparmos esses agentes em sociedades – de maneiras muito especiais – acabamos nos deparando com a verdadeira inteligência*". [MIN87]

Travers transporta esta concepção para o domínio da programação de computadores. Nesse contexto "agente" assume uma forma derivada da primeira. Os pequenos pedaços agora são feitos de software. Deste modo ele define agente como:

> *"[...] qualquer componente de um programa ou sistema que é desenhado para ser concebido como animado. O termo 'agente' sugere uma variedade de atributos, como determinação, autonomia, e a habilidade de reagir a estímulos externos, que não são geralmente embutidos nas metáforas que fundamentam a programação"*. [TRA96]

Esta concepção de agente é a que iremos adotar na maior parte do nosso trabalho, principalmente por ser este o direcionamento tomado pela maioria dos programas e metodologias que serão analisados.

Por outro lado, os agentes podem ser interpretados sob um prisma que indica uma direção diversa da perspectiva apresentada, ou seja, como entidades dotadas de inteligência. Seu grau de inteligência tem função essencial para caracterizar sua autonomia perante as situações que lhe são apresentadas.

> *"A autonomia de um agente requer inteligência devido à necessidade de sobreviver em um ambiente real, dinâmico e nem sempre benigno. Está intimamente relacionada com o tipo de arquitetura do sistema; assim, podemos ter agentes com diferentes graus de autonomia"*. [GIR99]

O limite entre as duas interpretações pode, em muitas situações, não ser tão claro e conduzir à questão do que se denomina como inteligência.

Minsky utiliza também o termo "agência" [MIN87] para descrever como diversos agentes se reúnem em grupos (sociedades) formando uma entidade que é capaz de realizar atividades mais complexas e dão a impressão, para quem as vê externamente, de que "sabem" o que estão fazendo.

Podemos então entender que ambas as concepções de agentes podem estar relacionadas, mesmo se adotam abordagens diferentes para o problema.

5.2.1. Objetos x Agentes

Em muitos aspectos os agentes se parecem com objetos e corremos o risco de toma-los como iguais.

Existem diferenças bastante claras entre ambos. Conforme será demonstrado, tomaremos os agentes como casos especiais de objetos dotados de características distintas.

Nicholas Jennings [JEN98] descreve algumas características distintas dos agentes[2]:

Autonomia – podemos afirmar que os agentes são dotados de níveis de autonomia maior quanto ao seu comportamento perante ao ambiente. De uma forma geral, as ações dos objetos estão centradas em respostas a invocações de seus métodos. Os agentes, por outro lado, também podem receber estímulos externos, tal como os objetos, no entanto, a decisão de qual atitude deve ser tomada é determinada pelo próprio agente. Existe deste modo uma diferença do centro de decisões e conseqüentemente da autonomia.
Isto não impede que os objetos sejam utilizados para a modelagem de agentes, pelo contrário, podemos implementar métodos que suportem decisões quanto a sua atuação, centrando as decisões no próprio objeto.

Comportamento – "A segunda importante distinção entre objeto e agente diz respeito à noção de comportamento autônomo e flexível (reativo, pró-ativo, social). O modelo de objetos padrão não tem nada para dizer sobre como construir sistemas que integram estes tipos de comportamento" [JEN98].

Processo Independente – cada agente está associado a um processo, que é executado de forma independente. Isto lhe dá autonomia e permite que múltiplos agentes atuem e interajam de forma concorrente.Apesar de existirem linguagens de programação orientadas a objetos que dão suporte a execução de tarefas concorrentes, em geral existe um processo único de controle central e, em alguns casos, podem ser disparados processos concorrentes.

Estas três características nos permitem distinguir a fronteira entre objetos e agentes. No entanto, como já foi apontado, podemos conceber agentes como casos particulares de objetos. Todas os aspectos aqui relacionados, apesar de não serem característicos de modelos baseados em objetos, podem ser implementados nos mesmos. Como trataremos adiante, esta possibilidade é especialmente interessante para a integração dos agentes com outros objetos educacionais.

Tal como os objetos, os agentes são metáforas que cumprem um papel de mediação entre um modelo mental humano e a verdadeira representação computacional deste modelo.

---

[2] No tópico "*2.1.2. Object and Concurrent Object Systems*", Nicholas Jennings e outros detalham os aspectos que distintivos entre objetos e agentes.

5.3. Autômatos Celulares

> "*Autômatos celulares são sistemas dinâmicos discretos cujo comportamento é completamente descrito em termos de relações locais. Um autômato celular pode ser pensado como um universo estilizado. O espaço é representado por um grid uniforme, com cada célula contendo alguns bits de dados; o tempo avança em passos discretos e as leis do universo são expressas em digamos , uma pequena tabela de referência, através da qual a cada passo as células computam seu novo estado a partir do próprio estado e do estado de seus vizinhos próximos*" [SMI98].

Um dos mais famosos autômatos celulares foi o "Jogo da Vida" criado por John Horton Conway em 1970 [SMI98]. Baseados em um conjunto de regras simples, e em células que podem assumir apenas dois estados (viva ou morta) ele mostrou que é possível se criar sistemas bastante complexos com comportamento semelhante à vida orgânica.

Sistemas de simulação tem adotado abordagens mistas entre autômatos celulares e objetos ou autômatos celulares e agentes. Os objetos e agentes são elementos que preenchem as células, como veremos adiante esta mixagem alia poderosos recursos de modelagem destas abordagem e tem dado origem a ambientes bastante propícios ao desenvolvimento de atividades de ensino aprendizagem.

6. Objetos Educacionais

6.1. Intercâmbio e distribuição de objetos

Uma parte do conteúdo produzido sobre a forma de objetos, especialmente os objetos-componentes, dispõem de uma independência e generalidade suficientes para permitir a sua customização e reaproveitamento em outras aplicações.

Tem se tornado prática, principalmente entre os desenvolvedores de *software*, o amplo intercâmbio de bibliotecas contendo classes de objetos.

Linguagens de programação como o Java, cujos programas podem ser executados em múltiplas plataformas e possui um padrão aberto para a montagem de componentes, tem promovido ainda mais este intercâmbio.

A Internet se tornou o principal meio por onde circulam estes objetos. Nela são disponibilizadas bibliotecas publicamente, e existem páginas que tem se especializado em realizar coletâneas destas bibliotecas, as categorizando conforme a aplicação.

Muitos objetos computacionais podem ser apresentados diretamente em um navegador *Web*, seja porque o navegador possui um módulo de software que o executa diretamente (*Virtual Machine* em Java, por exemplo), seja através de um programa que se acopla ao navegador dinamicamente (*Plugin*) para a execução do objeto.

Também na área educacional existem projetos que promovem a difusão de objetos, tal como o EOE – *Educational Object Economy*[3] que, em conjunto com o *Center for Distributed Learning* (CDL)[4], promovem um modelo para uma comunidade de aprendizagem onde circulam ativamente produções sob a forma de objetos. Como aponta Martin Koning-Bastiaan, desta maneira será possível tirar proveito do esforço de uma enorme quantidade de pequenos criadores de software educacional no mundo.

> *"Ambos o EOE e o CDL chegaram a uma compreensão sobre software educacional: pequenos objetos caem dentro das fendas. Outro ponto importante: muitos deles são pequenos objetos. Os grandes objetos encontram seu caminho em CD-ROMs ou são vendidos em pacotes courseware. Ainda é um sonho de muitos desenvolvedores-educadores que alguém de uma grande editora verá que seu produto vale a pena e pagará muito dinheiro por ele. A realidade, certamente, é que isto acontece com muito poucas peças de software educacional. A tragédia não está nisto, mas especialmente que os outros softwares raramente vêem a luz do dia novamente em outro lugar além da sala de aula do educador"*. [KON].

Com a expectativa de um grande crescimento deste intercâmbio, surge um outro desafio: criar um sistema eficiente e padronizado de classificação para esta enorme quantidade de objetos distribuídos por todo o mundo.

6.2. Metadados e Objetos de Conhecimento

Para atender a esta demanda emergente, o IEEE *Learning Technology Standardization Committee* (LTSC)[5] tem liderado um projeto para a elaboração de um padrão de metadados para "objetos de conhecimento", o LOM - *Learning Object Metadata* [LTS00].

Estes metadados são dados que descrevem os objetos de conhecimento, ou melhor dizendo, dados que descrevem outros dados (metadados) sob a forma de objetos de conhecimento.

Diversos esforços têm sido realizados para se estabelecer vocabulários comuns de metadados, através da padronização de conjuntos de descritores, interpretados de forma única entre as diversas comunidades. Estes padrões promovem a interoperabilidade entre sistemas e comunidades diversos. Em particular, pode ser destacada a iniciativa Dublin Core [DC97].

A Dublin Core constitui um conjunto de metadados com o objetivo original de permitir que autores agreguem uma descrição a seus recursos produzidos para *Web*. Esta descrição facilita a exploração de recursos eletrônicos. Tal iniciativa atraiu a atenção de comunidades que fazem uso de descrição formal de recursos como museus, bibliotecas, agencias governamentais, e organizações comerciais.

---

[3] **EOE** – *Educational Object Economy* – http://www.eoe.org/index.htm

[4] **CDL** – *Center for Distributed Learning* – http://www.cdl.edu/

[5] **LTSC** - http://ltsc.ieee.org/

Na área educacional as iniciativas na elaboração de conjuntos de metadados para uma estrutura descritiva de recursos educacionais são muitas, como o Projeto ARIADNE[6], o *Learning Object Metadata* (LOM), e o IMS *Learning Resource Meta-data* [AND00].

Os três projetos estão bastante inter-relacionados e se apóiam sobre a definição de Objeto de Conhecimento (*Learning Object*) definido pelo IEEE:

> *"Objetos de conhecimento são aqui definidos como qualquer entidade, digital ou não digital, que pode ser utilizada, reutilizada ou referenciada durante o aprendizado apoiado sobre a tecnologia. Exemplos de aprendizado apoiado sobre a tecnologia incluem sistemas de treinamento baseados no computador, ambientes de aprendizado interativo, sistemas inteligentes de instrução auxiliada por computador, sistemas de aprendizado a distância, sistema de aprendizado baseados na Web, e ambientes de aprendizado colaborativo"* [LTS00].

O termo "objeto" é utilizado de forma bem mais ampla que aquela estabelecida pelo paradigma da orientação a objetos, se referindo a entidades digitais e não digitais. De qualquer modo, por ser mais abrange se aplica igualmente ao modelo de objetos apresentado.

Os metadados que descrevem estes objetos são definidos em termos de propriedades e valores. Cada objeto possui um conjunto de propriedades a ele relacionadas como: assunto, data de criação, etc. Uma instância específica deste objeto possui valores para cada propriedade, por exemplo:

> Assunto = "Usando componentes"
> Data de criação = "15/07/2000"

Para padronizar as propriedades que descrevem cada objeto de conhecimento, no sentido de viabilizar o intercâmbio dos mesmos, a especificação adota o conceito de esquemas (*schemes*). Os esquemas são estruturas hierárquicas que determinam quais são as propriedades associadas a um objeto de conhecimento, definindo para cada um o seu tipo, domínio e obrigatoriedade.

O padrão do IEEE é montado sobre um esquema bastante genérico, denominado BaseScheme (esquema básico), cujo princípio é reunir os principais elementos comuns entre os objetos educacionais. Este esquema pode ser estendido para outros esquemas derivados de acordo com necessidades específicas.

A estrutura geral do BaseScheme é dividida segundo grupos de propriedades:

General	Dados de identificação, chaves de acesso, delimitação do domínio, descrição do conteúdo e da estrutura.
LifeCycle	Dados para controle e documentação do ciclo de vida do documento.

[6] **Projeto ARIADNE** – http://ariadne.unil.ch/

MetaMetaData	Referências à origem e estrutura dos metadados.
Technical	Dados técnicos, tais como: formato, tamanho, requisitos de sistema operacional, duração, etc.
Educational	Elementos de descrição pedagógica do recurso, tais como: abordagem, nível de interatividade, pré-requisitos, objetivo educacional, grau de dificuldade, tempo esperado de trabalho, etc.
Rights	Dados referentes às condições de uso do produto e eventualmente valores a serem pagos pelo uso do recurso.
Relation	Características do recurso em relação a outros recursos.
Annotation	Comentários referentes ao uso educacional do produto.
Classification	Referência que determina onde o recurso será colocado dentro de um sistema de classificação específico.

A estrutura da especificação de metadados do IMS é baseada no LOM, com algumas modificações. Adicionalmente, o IMS define uma representação em XML para os metadados.

6.2.1. XML e IMS

Em 1996, foi lançada a proposta da XML (*Extensible Markup Language*) [W3X98], como uma versão simplificada da SGML (*Standard Generalized Markup Language*) [ISO86] para ser utilizada na WWW. Ao contrário da HTML, XML não é uma linguagem escrita em SGML, mas uma linguagem derivada da mesma, isto significa que ela possui recursos que permitem a uma comunidade definir seus próprios marcadores, as regras de composição de documentos e, conseqüentemente, a atribuição de significado particular aos segmentos marcados. Isto é possível pois SGML é uma meta linguagem (linguagem para a definição de linguagens).

A linguagem XML tem dado origem a uma série de outras linguagens, tal como a SMIL (*Synchronized Multimedia Integration Language*) [W3S98], utilizada para elaboração de documentos multimídia, e o MathML (*Mathematical Markup Language*) [W3C99], voltada para a representação de documentos de conteúdo matemático e científico.

XML se insere na família das linguagens de marcação. Utilizando marcadores, estas linguagens têm a capacidade de agregar informações adicionais a documentos com os mais diversos propósitos. No caso de XML os marcadores são caracterizados pelos símbolos “<” e “>” (seguem o mesmo princípio do HTML). Deste modo a expressão:

```
<sujeito>O dinossauro </sujeito> <predicado>atravessou o rio</predicado>
```

Permite diferenciar o conteúdo da frase “O dinossauro atravessou o rio” das marcações (<sujeito> e <predicado>) que agregam informações adicionais à frase, neste caso identificando seu sujeito e predicado (para mais detalhes sobre XML vide Anexo 1).

Para mapear metadados de objetos de conhecimento para o formato XML, o IMS adota o mesmo modelo do LOM (par propriedade / valor) utilizando, em geral, o marcador para

representar a propriedade, e o conteúdo entre os marcadores representando o valor da propriedade. Desta maneira a sentença:

DATETIME = "2000-11-08"

É escrita desta maneira:

```
<DATATIME>2000-11-08</DATETIME>
```

Por ser o XML um padrão aberto, extensível e adequado para *Web*, sua combinação com o LOM, como propõe o IMS, resulta em um poderoso sistema aberto para se agregar metadados a objetos de conhecimento.

Por outro lado, o IMS teve que desenvolver sobre o XML seu próprio mecanismo para o mapeamento dos metadados definidos no LOM, além disto, definiu uma especificação para associar os metadados ao objeto de conhecimento.

Fazendo assim, o IMS criou uma metodologia particular de associação de metadados a recursos. No entanto, esta metodologia poderia ser mais aberta e abrangente se fizesse uso de algum padrão estabelecido para metadados estruturados.

6.2.2. RDF

RDF - *Resource Description Framework* - é uma linguagem baseada em XML destinada à codificação, troca e reutilização de metadados estruturados.

A idéia básica de RDF [LAS99] não é definir um conjunto universal de metadados e sim prover os mecanismos necessários para que as diversas comunidades codifiquem, troquem e reutilizem metadados estruturados.

Tomemos uma classe de objetos, codificada em linguagem Java, e armazenada em um arquivo de nome "Molecula.class". O autor desta classe se chama Jorge.

Queremos representar esta relação através de RDF. Isto significa, na terminologia desta linguagem, acrescentar o valor de uma propriedade na descrição do recurso “Molecula.class”.

Para descrever um recurso, RDF utiliza instruções compostas de três partes: recurso, tipo de propriedade e valor.

[Molecula.class] --- autor ---> “Jorge”

Esta instrução escrita em RDF se apresenta como:

```
<RDF:Description about = "Molecula.class">
  <EDUC:autor>Jorge</EDUC:autor>
</RDF:Description>
```

As facilidades promovidas pelo RDF para se agregar metadados a recursos de forma padronizada, além de outras, tal como a capacidade de definir esquemas (*scheme*) e de trabalhar com classes e herança, o tornam bastante apropriado como padrão para metadados de objetos de conhecimento (para mais detalhes do RDF vide Anexo 2). O BaseScheme definido pelo LOM, por exemplo, pode ser codificado através de um esquema (*scheme*) RDF.

Por este motivo, apesar de desconhecermos alguma proposta neste sentido, acreditamos ser esta uma tendência futura.

6.3. Objetos, simulação e aprendizagem ativa

> *"Hoje em dia, a simulação exerce um papel crescente nas atividades de pesquisa científica, de concepção industrial, de gestão, de aprendizado, mas também para o jogo e a diversão (em especial os jogos interativos na tela). Em teoria, em experiência, a maneira de industrialização da experiência de pensamento - a simulação - é um modo especial de conhecimento, próprio da cybercultura nascente. Na pesquisa, seu principal interesse não está, evidentemente, na substituição da experiência, nem em fazer as vezes de realidades, mas em permitir a formulação e a rápida exploração de um grande número de hipóteses. Sob o ângulo da inteligência coletiva, ela permite a colocação em imagens e a partilha de mundos virtuais e de universos de significado de uma grande complexidade"*. [LEV98]

Ambientes baseados em objetos e especialmente aqueles baseados em agentes, são um território bastante promissor para a elaboração de simulações com propósitos educacionais. Nestes ambientes, o aprendiz adota uma postura ativa na construção de modelos de simulação, sua experimentação e refinamento. Dentro deste processo, ele vai se tornando artífice do próprio conhecimento.

> *"Em um relatório sobre tecnologia educacional para o Presidente dos Estados Unidos, um comitê de consultores de ciências apresentaram as mais promissoras aplicações construtivistas da tecnologia, com simulações no topo da lista"*. [REP99]

6.4. Componentes Educacionais

Muitos dos atuais ambientes e sistemas, projetados para uso em atividades educacionais, perceberam a importância de estruturar suas peças de software sob a forma de componentes. Este tipo de estrutura pode trazer diversos benefícios.

Do ponto de vista do professor / autor do software educacional:

- Os componentes por ele produzidos podem ser combinados, ou recombinados, com outros, não limitando sua produção a um programa específico.
- Componentes são facilmente configuráveis e podem ser customizados a necessidades particulares.

- Os dois fatores anteriores, aliados à facilidade de transportar componentes como pacotes autônomos e integráveis, promovem amplamente o intercâmbio dos componentes de autores diversos, e a Internet facilita e potencializa este intercâmbio.
- Não será necessário partir da estaca zero em cada nova produção, pois os componentes criados podem ser reutilizados, além dos componentes produzidos por terceiros.

Do ponto de vista do aluno:

- Utilizar componentes como blocos de construção promovem o desenvolvimento de atividades onde o aluno atua como agente ativo na construção do conhecimento.
- O intercâmbio facilita a colaboração entre alunos dentro de um mesmo grupo ou entre grupos. Aqui a Internet também cumpre um papel importante de mediadora para uma comunicação que pode se estender a um âmbito global. Além disto, ela também cumpre o papel de repositório de componentes.

Até mesmo os sistemas baseados em agentes, os estruturam dentro de componentes. Como já foi abordado anteriormente, isto é possível dada a ligação comum de ambas as abordagens, agentes e componentes, com o modelo de objetos.

6.5. Ambientes e Sistemas Educacionais

Existe atualmente uma grande diversidade de ambientes e sistemas cujas estruturas estão baseadas em modelos de objetos. Muitos deles, mesclam outros modelos apresentados neste trabalho (componentes, agentes e autômatos celulares). Por exemplo, grande parte dos sistemas baseados em agentes faz uso de um espaço e tempo discreto, fundamentado nos autômatos celulares. Um dos benefícios desta mixagem é a simplicidade em que se podem dispor os agentes, e estabelecer suas regras de comportamento e interação.

Por este motivo, a análise dos ambientes e sistemas não deve se fundamentar na classificação de um único modelo adotado, mas na influência de diversos modelos e os benefícios extraídos de cada um.

A seguir apresentaremos um resumo de alguns dos ambientes e sistemas que utilizam o modelo de objetos, e que serviram como base prática para todo o estudo apresentado.

6.5.1. Casa Mágica

O sistema Casa Mágica [SAN98] constitui-se em um ambiente para construção de aplicações educacionais que combina os recursos de uma ferramenta de autoria com recursos que permitem a construção e exploração de modelos de estudo.

Casa Mágica oferece suporte para a elaboração de *frameworks* baseadas em objetos e componentes e é capaz de representá-los em XML [SAN00]. Assim, as aplicações podem ser distribuídas pela Internet e executadas diretamente por um módulo *runtime* do sistema ou apresentadas em um navegador *Web*.

6.5.2. E-Slate

O E-Slate[7] permite a construção de micro-mundos utilizando uma biblioteca de componentes educacionais especialmente projetada para que sejam facilmente interligados. Tais componentes podem ser diretamente combinados em páginas *Web* e seu comportamento pode ser descrito através de uma linguagem de script baseada no LOGO.

6.5.3. MathWorlds

O MathWorlds, software integrante do projeto SimCalc[8] *"[...] provê uma coleção de componentes de software, incluindo um conjunto de mundos de animação e uma variedade de gráficos. Atores nos mundos (como um palhaço, ou um pato) movem-se de acordo com funções matemáticas. Gráficos mostram estas funções matemáticas e permitem aos alunos editarem diretamente as funções"*. [RSC98]

Sua estrutura é bastante coerente com o objetivo do projeto SimCalc, que pretende combinar a tecnologia das simulações com um currículo inovador, iniciado nas primeiras séries e se estendendo para além do cálculo clássico.

6.5.4. Homos

Este software é destinado a simulações com propósitos educacionais e tem como base os autômatos celulares. Em cada célula pode ser colocado um objeto, cuja classe é definida pelo usuário.

O comportamento dos objetos é determinado por um conjunto de regras associadas à classe dos mesmos. O resultado das simulações pode ser traçado em gráficos.

6.5.5. LiveWorld

Trata-se de um ambiente de programação para sistemas animados baseados em agentes. O LiveWorld é parte do trabalho "Programming with Agents: New metaphors for thinking about computation" de Michael Travers [TRA96] e serve como referencial para análises, feitas por ele, referentes à metáfora dos agentes na programação.

6.5.6. Agentsheets e Stagecast

Estes dois softwares são baseados em agentes cuja atuação se desenvolve em espaço e tempo discretos. Sua abordagem é especialmente adequada para a realização de atividades pedagógicas que usam simulação.

O Agentsheets possui o **Visual Agent Talk** [REP96], que consiste em um ambiente para intercâmbio de elementos de *software* que usam a *Web* como meio. Esse ambiente possibilita o desenvolvimento de atividades colaborativas entre alunos, em que a *Web* funciona como uma biblioteca na qual os elementos podem ser guardados e recuperados por qualquer equipe.

---

[7] **E-Slate:** http://e-slate.cti.gr/

[8] **SimCalc** - http://tango.mth.umassd.edu/

6.5.7. Alice

O Alice consiste em um ambiente tridimensional para a construção de animações com propósitos educacionais.

Sua estrutura para organização dos modelos tridimensionais e montagem das animações é toda orientada a objetos. Um modelo tridimensional é considerado um objeto dividido em partes, que também são objetos. O movimento dos objetos é feito através da ativação de métodos dos mesmos.

6.6. Integração de Objetos

Sistemas como o MathWorlds, E-Slate, AgentSheets e JavaSketchpad[9] estão inseridos no projeto ESCOT [ROS98], que tem estudado meios de realizar a integração destes diferentes sistemas, baseando-se no modelo de componentes intercomunicantes.

O ESCOT elegeu Java como plataforma de integração dos componentes. Muitos sistemas já utilizam Java, tal como o MathWorlds e o E-Slate. Outros, como o AgentSheets e o Sketchpad[10], desenvolveram conversores que transformam suas produções em coleções de componentes Java. Este último assume o nome de JavaSketchpad.

Através do ESCOT, tem sido desenvolvidas atividades onde os alunos integram componentes de diversas origens. Uma simulação do AgentSheets, por exemplo, pode ser interligada a um gráfico do MathWorlds. Deste modo, os alunos podem construir a simulação da difusão de um vírus em uma comunidade e conecta-la com um componente que exibe graficamente a curva de progressão da doença [REP96].

7. Considerações Finais

Na elaboração deste trabalho, foi feita uma seleção de um conjunto de ambientes e sistemas a ser analisados, de tal forma que pudessem ilustrar o papel fundamental que a tecnologia de objetos tem cumprido em atividades de ensino-aprendizagem fazendo uso da tecnologia.

Ao optarmos por uma seleção, muitas outras abordagens bastante interessantes e significativas não puderam ser aqui analisadas. Isto não indica sua menor importância ou significação dentro do panorama apresentado. Pelo contrário, indica que existem ainda muitas perspectivas bastante ricas do uso de objetos na educação, não cobertas neste trabalho.

Ao final, não nos encontramos, portanto, em um ponto de chegada, mas consideramos as análises, aqui apresentadas, um referencial de partida para um futuro estudo bem mais amplo.

---

[9] **JavaSketchpad** - http://www.keypress.com/sketchpad/java_gsp/

[10] **Geometer's Sketchpad** – consiste em um ambiente dinâmico popular de exploração matemática.

Referências Bibliográficas

[AND00] **Anderson, Thor & Wason, Tom**. IMS Learning Resource Meta-data Information Model – Version 1.1 - Final Specification. IMS Global Learning Consortium, Inc., 5 June 2000, [Online] http://www.imsproject.com/metadata/mdinfov1p1.html

[BRA98] **Bray, Tim & Hollander, Dave & Layman, Andrew**. Name Spaces in XML - World Wide Web Consortium Note 19-January-1998, W3C - World Wide Web Consortium, [Online] http://www.w3.org/TR/1998/NOTE-xml-names-0119

[BRI99] **Brickley, Dan & Guha, R. V.** Resource Description Framework (RDF) Schema Specification - W3C Proposed Recommendation 03 March 1999, W3C - World Wide Web Consortium, [Online] http://www.w3.org/TR/PR-rdf-schema/

[DC97] **Dublin Core**. The Dublin Core: A Simple Content Description Model for Electronic Resources, February 1997, [Online] http://purl.oclc.org/dc/index.htm.

[FAR99] **Farance, Frank**. LOM Specification - Learning Objects Metadata, Proposed Draft 4. IEEE Learning Technology Standards Committee (LTSC), September, 1999, [Online] http://edutool.com/lom/lom-04.pdf

Este é o documento oficial onde está a especificação do LOM, elaborada pelo Learning Technology Standards Committee (LTSC) do IEEE.

[FER89] **Ferreira, Aurélio B. H.** Minidicionário da Língua Portuguesa. Rio de Janeiro, Editora Nova Fronteira, 1989.

[HOL92] **Holmevik, Jan Rune**. Compiling SIMULA: a historical study of technological genesis. IEEE Annals of the History of Computing, 16 (4), p. 25-37, 1994. [Online] http://www.cis.um.edu.mt/~jskl/simula.html

[IBM] **IBM - International Business Machines Corporation**. IBM Smalltalk Tutorial [Online] http://www.wi2.uni-erlangen.de/sw/smalltalk/

[ING81] **Ingalls, Daniel H. H.** Design Principles Behind SmallTalk. BYTE Magazine, August 1981 [Online] http://users.ipa.net/%7edwighth/smalltalk/byte_aug81/design_principles_behind_smalltalk.html

Daniel H. H. Ingalls chefiou a implementação das primeiras versões do Smalltalk. Neste artigo ele apresenta os fundamentos desta linguagem que são a primeira expressão concreta da Programação Orientada a Objetos. Aqui são lançadas as bases que nortearão as futuras linguagens e modelos.

[ISO86] **ISO - International Organization for Standardization.** ISO 8879:1986(E), Information processing - Text and Office Systems - Standard Generalized Markup Language (SGML), First edition - 1986-10-15. Geneva: International Organization for Standardization, 1986.

[JEN98] **Jennings, Nicholas R., Sycara, Katia and Wooldridge, Michael**. A Roadmap of Agent Research and Development. In: Autonomous Agents and Multi-Agent Systems Journal. Boston: Kluwer Academic Publishers, 1998, Volume 1, Issue 1, pages 7-38 [Online] http://www.ri.cmu.edu/pub_files/pub1/jennings_nicholas_1998_1/jennings_nicholas_1998_1.pdf

[KON] **Koning-Bastiaan, Martin**. Connected and Scalable: A revolutionary Structure for Online Communities. EOE – *Educational Object Economy* [Online] http://www.eoe.org/FMPro?-db=Objects.fp3&-token=libraryPapers&-format=/library/paperdetail.htm&-recid=35433&-lay=all&-Find

[KRE98] **Kreutzer, Wolfgang**. Basic Aspects of Squeak and the Smalltalk-80 Programing Language. Junho, 1998 [Online] http://kaka.cosc.canterbury.ac.nz/~wolfgang/cosc205/smalltalk1.html

[KUH87] **Kuhn, Thomas.** "Posfácio - 1969", in A estrutura das revoluções científicas. São Paulo: Perspectiva, 1987.

[LEV98] **Lévy, Pierre**. Educação e Cybercultura – A nova relação com o saber. 1998 [Online] http://www.portoweb.com.br/PierreLevy/educaecyber.html

[LTS00] **IEEE Learning Technology Standards Committee (LTSC)**. Draft Standard for Learning Object Metadata. Institute of Electrical and Electronics Engineers, Inc., 5 February 2000, [Online] http://ltsc.ieee.org/doc/wg12/LOM_WD4.PDF

[GIR99] **Giraffa, Lucia Maria Martins**. Uma arquitetura de tutor utilizando estados mentais. Porto Alegre: CPGCC/UFRGS, 1999. (Tese de Doutorado) [Online] http://www.inf.pucrs.br/~giraffa/ia/tese.zip

[LAK80] **Lakoff, George and Johnson, Mark**. Metaphors We Live By. Chicago: University of Chicago Press, 1980.

[LAS99] **Lassila, Ora & Swick, Ralph R.** Resource Description Framework (RDF) Model and Syntax Specification - W3C Recommendation 22 February 1999, W3C - World Wide Web Consortium, [Online] http://www.w3.org/TR/REC-rdf-syntax/

[MAN97] **Manola, Frank** (ed.). NICTS Technical Committee H7 Object Model Features Matrix, X3H7-93-007v12b, May 25, 1997 [Online] http://www.objs.com/x3h7/h7home.htm.

[MAN98] **Manola, Frank**. Towards a Web Object Model. Object Services and Consulting, Inc. (OBJS),10 February 1998 [Online] http://www.objs.com/OSA/wom.htm

[MAN99] **Manola, Frank.** Technologies for a Web Object Model. IEEE Internet Computing, January - February 1999.

[MEY97] **Meyer, Bertrand**. Object-Oriented Software Construction – Second Edition. USA, Prentice-Hall, Inc., 1997.

Realiza uma análise bastante aprofundada da construção de software orientado a objetos. Neste livro são analisados as características de linguagens orientadas a objetos, sua estrutura e conceitos.

[MIL98] **Miller, Eric**. An Introduction to the Resource Description Framework, May 1998, [OnLine] http://www.dlib.org/dlib/may98/miller/05miller.html

[MIN87] **Minsky, Marvin**. Society of Mind. New York: Simon & Schuster, 1987.

[REP96] **Repenning, Alexander & Ambach, James.** Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing. Proceedings of the 1996 IEEE Symposium of Visual Languages, Boulder, CO, Computer Society, 1996, pp. 102-109 [OnLine] http://www.cs.colorado.edu/~l3d/systems/agentsheets/Documentation/VL-96-Paper.pdf

[REP99] **Repenning, Alexander & Ioannidou , Andri & Phillips, Jonathan**. Collaborative Use & Design of Interactive Simulations. Proceedings of Computer Supported Collaborative Learning Conference at Stanford (CSCL'99), 1999 [Online] http://www.escot.org/escot/External/UseDesign.pdf

[ROS98] **Roschelle, Jeremy & Digiano, Chris & Pea, Roy & Kaput, Jim.** Educational Software Components of Tomorrow (ESCOT). SRI International, 1998, [Online] http://www.escot.org/escot/External/MSET_ESCOT.html

[RSC98] **Roschelle, Jeremy & Kaput, Jim & Stroup, W.** SimCalc: Accelerating students' engagement with the mathematics of change. In M. Jacobson & R. Kozma (Eds.), Educational technology and mathematics and science for the 21st century. Hillsdale, NJ: Erlbaum, 1998 [Online] http://tango.mth.umassd.edu/website/publications/MathofChange.pdf.

[SAN98] **Santanchè, André.** Sistema para Construção de Aplicações Educacionais. IV Congresso Ibero-Americano de Informática na Educação, 1998, [Online] http://www.brasil.terravista.pt/claridade/1622/publicado/ArtigoRIBIE1998.pdf

[SAN00] **Santanchè, André & Teixeira, Cesar**. Construindo e explorando o conhecimento através de Componentes Educacionais embutidos em hiperdocumentos. VI Workshop de Informática na Escola – XX Congresso da

SBC, 2000, [Online]
http://www.brasil.terravista.pt/claridade/1622/publicado/WIE2000.pdf

[SMI98] **Smith, Sophia**. Artificial Life and Genetic Algorithms. Brunel University, Artificial Intelligence Site, Abril de 1998 [Online]
http://www.brunel.ac.uk:8080/depts/AI/alife/alife-main.html

[TRA96] **Travers, Michael D.** Programming with Agents: New metaphors for thinking about computation. Massachusetts Institute of Technology, June 1996 [Online]
http://lcs.www.media.mit.edu/people/mt/thesis/mt-thesis.html

[W3C99] **W3C XML Working Group.** Mathematical Markup Language (MathML™) 1.01 Specification - W3C Recommendation, revision of 7 July 1999, W3C - World Wide Web Consortium, [Online] http://www.w3.org/TR/REC-MathML/

[W3S98] **W3C XML Working Group.** Synchronized Multimedia Integration Language (SMIL) 1.0 Specification - W3C Recommendation 15-June-1998, W3C - World Wide Web Consortium, [Online] http://www.w3.org/TR/REC-smil/

[W3X98] **W3C XML Working Group**. Extensible Markup Language (XML) 1.0 - W3C Recommendation 10-February-1998, W3C - World Wide Web Consortium, [Online] http://www.w3.org/TR/1998/REC-xml-19980210

Anexo 1

Um pouco mais de XML

XML estabelece um mecanismo de marcação de conteúdo semelhante ao HTML. Tal mecanismo se baseia na inserção de marcadores que permitem a atribuição de significados a trechos de um documento. Todos os marcadores são delimitadas pelos símbolos "<" e ">".

Por exemplo, podemos estabelecer um marcador que delimita o título de um documento denominado <Titulo>. Desta forma, sua marcação se apresentaria da seguinte maneira:

```
<Titulo>Usando componentes</Titulo>
```

As marcações podem ser agrupadas hierarquicamente e podem mudar sua interpretação a partir do contexto. Tomemos como exemplo um artigo cujo título seja “Usando componentes”, dividido em tópicos com subtítulos. Cada tópico será delimitado por um marcador <Topico>:

```
<Titulo>Usando componentes</Titulo>
<Autor>Jorge</Autor>
<Topico>
            <Titulo>Introdução</Titulo>
            :
            :
</Topico>
```

Para se estabelecer o título do tópico (subtítulo), foi usado o mesmo marcador <Titulo>, no entanto, por estar dentro de <Topico> será interpretado como um subtítulo associado ao tópico, e não como título do artigo.

Os marcadores que delimitam um conteúdo possuem uma marcação de início e uma de final, cujo identificador é precedido por uma barra. XML permite a criação de marcadores vazios encerrados por uma barra (/). Por exemplo, o marcador:

```
<Separador/>
```

Por ser encerrado por uma barra, não possui conteúdo e conseqüentemente não possui marcador de fim.

Os marcadores associados ao conteúdo de documentos XML são denominados elementos. Os elementos podem possuir atributos, definidos dentro do marcador. Cada atributo possui um valor a ele associado, como exemplificado na seguinte sentença:

```
<Documento ID = “doc001” data = “10/02/1997”>
```

No exemplo o elemento possui dois atributos: ID (identificador) e data.

Assim como o HTML, XML permite a constituição de ligações (*links*) para outros documentos (*links* externos) ou dentro do próprio documento (*links* internos).

Os *links* são estabelecidos a partir de um URI (*Uniform Resource Identifier*) que localiza de forma única um determinado recurso.

A identificação de fragmentos (sub-recursos) dentro de um documento HTML, é feita a partir da marcação prévia de alvos de localização. Para se fazer referência a este alvo é anexado o símbolo # depois do URI, seguido do identificador do alvo. No exemplo:

```
http://www.dominio.org/documento.html#bibliografia
```

Faz referência a um local marcado com o rótulo de "bibliografia" dentro de "documento.html".

HTML se limita no entanto a marcar um ponto dentro do documento sem, no entanto, estabelecer a extensão do fragmento. XML utiliza o símbolo # da mesma forma que HTML, porém, se o documento referenciado for XML, ao invés de fazer referência a um alvo demarcado, a identificação que segue o símbolo #, especifica um determinado elemento a ser extraído do documento. Como os elementos têm uma extensão precisamente delimitada, a partir de seus marcadores de início e final, é possível se definir exatamente o fragmento desejado.

Anexo 2

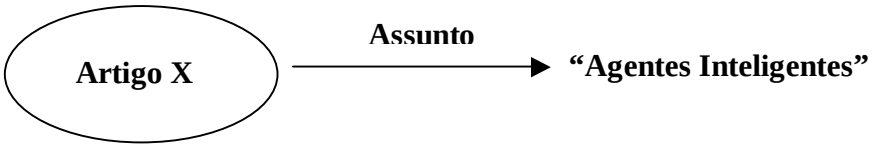
Um pouco mais de RDF

RDF define mecanismos para se associar propriedades a determinados recursos. Um recurso em RDF consiste de qualquer objeto que pode ser identificado a partir de uma URI, como por exemplo, um artigo X sobre Agentes Inteligentes, disponível em um documento no endereço: http://www.dominio.org/ArtigoX.txt (URI).

Ao recurso "Artigo X", pode ser associada uma propriedade "Assunto" que determina o assunto tratado no artigo. Considerando-se que o assunto seja “Agentes Inteligentes”, este será o valor da respectiva propriedade.

A tripla Recurso, Propriedade e Valor é a base que estabelece um modelo de dados em RDF. Tal modelo pode ser tanto compreendido por seres humanos, na medida em que este pode ser lido como: “O assunto do artigo X é Agentes Inteligentes”, como por máquinas que tem acesso a uma representação formal deste modelo.

A relação pode ser representada pelo diagrama abaixo:



Um mesmo documento pode fazer uso de vocabulários, para descrição de seu conteúdo, provenientes de mais de uma comunidade. Através do mecanismo de *Namespaces* do XML [BRA98], isto é possível em RDF. Cada comunidade estabelece seu esquema (*scheme*) específico, definido em um URI. Nos documentos onde este esquema será utilizado, realiza-se uma associação do URI a um prefixo ligado ao esquema. Veja o exemplo a seguir:

```
<?xml version="1.0"?>
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix ="RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>
<RDF:RDF>
      <RDF:Description about= "http://www.dominio.org/ArtigoX.txt">
            <DC:Suject>Agentes Inteligentes</DC:Subject>
      </RDF:Description>
</RDF:RDF>
```

Neste exemplo, pode ser observado o uso de dois vocabulários: o RDF e o DC. Os marcadores de cada um deles podem ser claramente identificados através dos prefixos RDF: e DC: respectivamente, que no início do documento são associados aos seus URIs. Isto permite a coexistência de diversos vocabulários, que determinam domínios diversos de metadados.

Cada vocabulário em RDF recebe o nome de *Schema*. Para se descrever um determinado recurso como um artigo, por exemplo, são utilizados atributos descritivos, tal como: autor, assunto, etc. A declaração destas propriedades (atributos) e da sua semântica correspondente são definidos no contexto do RDF como um RDF *schema* [BRI99].

Além de definir o conjunto de propriedades de um recurso, o *schema* provê meios de definir tipos de recurso, como por exemplo, páginas *Web*, artigos, livros, etc. Esta definição utiliza princípios de orientação a objetos. Os tipos são definidos em termos de classes e os recursos como instâncias destas classes (objetos). É possível se estabelecer subclasses que herdam as definições das superclasses.

As propriedades podem pertencer a uma classe e também podem ser do tipo de uma classe. Isto é bastante conveniente quando se pretende identificar atributos de acordo com uma determinada classe.