Anima: Sistema para integração de Objetos Educacionais

(Artigo publicado no SBIE2000 – Simpósio Brasileiro de Informática na Educação – 8 a 10 de novembro de 2000)

André Santanchè
UNIFACS[1] / NUPERC[2] / NUPPEAD[3]
santanche@unifacs.br

Cesar Augusto Camillo Teixeira
UNIFACS[1] / NUPERC[2]
cesar@unifacs.br

Resumo

O paradigma da Orientação a Objetos começa a extrapolar o domínio específico da computação e repercutir também na área de educação, não apenas no processo de elaboração dos softwares educacionais, mas também como metodologia subjacente no desenvolvimento de atividades de ensino-aprendizagem e como modelo na organização e distribuição de mídias educacionais.

Este trabalho apresenta um modelo referencial, utilizado para integrar objetos educacionais de diferentes linguagens e implementações, bem como um conjunto de mecanismos e ferramentas que permite a construção de *frameworks* com estes objetos, utilizadas como base para o desenvolvimento de atividades de ensino-aprendizagem, onde o aluno pode atuar como agente ativo na construção do conhecimento e trabalhar colaborativamente com outros alunos.

Palavras Chave

Metodologias para Desenvolvimento de Software Educativo, Objetos Educacionais, Componentes Educacionais, Metadados, Informática e Educação, Casa Mágica, *Anima*.

1. Introdução

A Programação Orientada ao Objeto, que teve seus princípios inicialmente materializados pela linguagem Simula e em seguida, de uma forma mais ampla, pela linguagem Smalltalk [ING81], trouxe grandes benefícios à computação. Entretanto, de um modo geral, tem ficado restrita ao domínio de especialistas desta área.

Mais do que uma técnica de programação, a Orientação a Objetos é constituída de um conjunto de princípios e um modelo de organização de idéias que se mostra igualmente aplicável em outras áreas da computação, como a análise e projeto de sistemas, modelagem de bancos de dados, etc. Deste modo a Orientação a Objetos conforma-se como um paradigma que modificou gradativamente a postura dos profissionais frente à computação e ao desenvolvimento de programas.

Nos últimos anos podemos observar que os objetos têm gradativamente retomado parte do propósito original do projeto Smalltalk[4], pelo menos no que diz respeito à educação. Tais objetos, especialmente sob a forma de componentes, se inserem em atividades de ensino-aprendizagem como blocos fundamentais, sobre os quais alunos e professores constroem modelos que se tornam alvo de estudo. Algumas destas experiências e uma revisão bibliográfica são relatadas no tópico **Objetos e a Educação**. Em paralelo, muitos esforços tem sido realizados no sentido de direcionar os padrões nativos da Internet para uma perspectiva "Orientada ao Objeto", o que se traduz em um poderoso modelo de objetos

[*] Este trabalho contou com o apoio do CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil
[1] **UNIFACS** – Universidade Salvador
[2] **NUPERC** - Núcleo de Pesquisa em Redes de Computadores
[3] **NUPPEAD** – Núcleo de Pesquisa e Projetos em Educação a Distância
[4] 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.

intercomunicantes através da rede e de objetos que trafegam pela rede. A linguagem XML - *eXtensible Markup Language* [W3C98], que será abordada no tópico **XML, RDF e XSL**, tem cumprido um papel importante neste contexto. Em conjunto com ela, RDF - *Resource Description Framework* [LAS99] - se apresenta como padrão para a associação de metadados a recursos e XSL - *eXtensible Stylesheet Language* [ADL00] - como um mecanismo para conversão dos documentos XML.

O tópico **Modelo Integrado de Objetos,** que constitui-se a contribuição mais relevante deste trabalho, apresenta o sistema *Anima*[5]. O sistema consiste em um modelo integrado para representação genérica de objetos, especialmente aqueles voltados para atividades de ensino-aprendizagem, suas interfaces, relações e metadados, segundo a proposta do LTSC[6] [FAR99], e de mecanismos e ferramentas que permitam a exploração de *frameworks* construídas segundo o modelo.

2. Objetos e a Educação

Estabelecer um panorama da atuação de objetos, em suas múltiplas perspectivas, na educação não é certamente um trabalho simples que se esgota no escopo de um artigo. O primeiro desafio é estabelecer claramente que tipo de influência o paradigma da orientação a objetos tem realizado em atividades de ensino-aprendizagem e em que aspectos.

De um lado os objetos podem ser encontrados atuando segundo a proposta original do Smalltalk, como artefatos de software que mediam a atividade criativa de usuários, tal como professores e alunos, e o computador. Os objetos se configuram como uma metáfora que representa o computador, seus algoritmos e dados. Tal metáfora é bem mais próxima da forma como percebemos o mundo [MEY88]. Como afirma Daniel Ingals: "*nós iremos poupar tempo se tornarmos nossos computadores compatíveis com a mente, ao invés do caminho contrário*". [ING81]

Novas metáforas, que partem de abstrações da percepção humana, transportadas para o domínio da computação, são poderosos mediadores para usuários não especialistas nesta área. O usuário modela seu projeto utilizando uma linguagem e percepção inerentes ao seu domínio e a máquina realiza o trabalho de mapear o modelo para a linguagem computacional.

2.1. Agentes

A metáfora dos agentes é um exemplo claro deste conceito. O termo "agente" foi utilizado por Marvin Minsky em seu livro "*Society of Mind*" [MIN87] para descrever pequenas unidades autônomas que realizam tarefas simples, mas que trabalhando em conjunto desenvolvem comportamentos mais complexos.

Michael Travers propõe a programação baseada em agentes [TRA96], que os utiliza como metáfora para a construção de sistemas animados. A partir desta metáfora foi concebida a ferramenta de autoria AgentSheets [REP96] para a construção de simulações.

Nesta ferramenta os alunos constroem cada agente separadamente e, através de uma linguagem visual, definem o comportamento dos mesmos. Estes agentes atuam dentro de um cenário e interagem entre si. Apenas ao se disparar o processo de simulação é possível verificar o comportamento do sistema completo, que é resultado do produto entre os comportamentos individuais dos agentes e a interação entre eles.

Atividades de ensino-aprendizagem como estas podem assumir um caráter fortemente construtivista, na medida em que os alunos se colocam em uma atitude ativa de construção do conhecimento.

---

[5] **Anima** – em latim significa "alma".

[6] **Learning Technology Standards Committee** - grupo de trabalho do IEEE que lidera os trabalhos para estabelecer uma sintaxe e semântica de metadados para Objetos de Aprendizado (LOM - Learning Objects Metadata).

Existe uma forte relação entre a representação de agentes e de objetos. Na verdade, os agentes podem ser modelados como tipos especiais de objetos dotados de autonomia e capazes de interagir com outros objetos agentes. Isto é especialmente interessante para a integração dos agentes com outros objetos educacionais.

2.2. 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. Por este motivo as linguagens de programação os tem configurado como um subconjunto específico de objetos, ainda que esta não seja uma relação obrigatória. Isto pode ser notado em linguagens dotadas de componentes de software sem, no entanto, serem orientadas ao objeto.

Os componentes se destacam por sua facilidade de configuração quando utilizados na construção de aplicações e podem ser interligados com outros, sem que o usuário saiba como eles foram implementados. Em geral, os sistemas de construção de aplicações que utilizam componentes dispõem de interfaces visuais para a instanciação, configuração e interligação dos componentes.

Construir uma aplicação usando componentes é semelhante a um modelo montado com peças de Lego®, que se encaixam para formar um modelo maior. A comunicação entre os componentes se dá através do envio de mensagens ou chamada de métodos.

Isto pode ser observado no exemplo apresentado na figura 1:

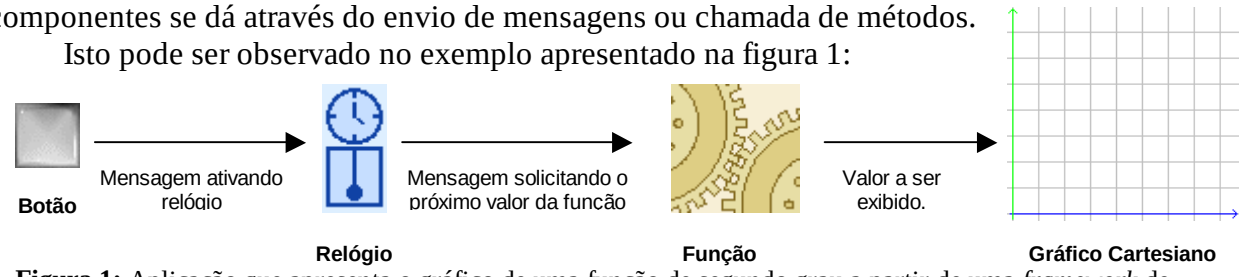

Figura 1: Aplicação que apresenta o gráfico de uma função de segundo grau a partir de uma *framework* de componentes

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 mapeia os valores recebidos.

Este conjunto de componentes interligado forma uma *framework*. A partir de bibliotecas de componentes especializadas em determinado domínio e de ferramentas visuais que permitem a customização e interligação de componentes, é possível que usuários leigos construam modelos mais complexos, trabalhando em um nível de abstração que desconhece os detalhes de implementação dos componentes.

Deste modo, componentes tornaram-se a peça fundamental de muitos projetos de ensino-aprendizagem. O MathWorlds, software integrante do projeto SimCalc, por exemplo, *"[...] 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]

Outro exemplo é o E-Slate[7], que permite a construção de micro-mundos utilizando uma

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

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.

As iniciativas acima descritas estão inseridas 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[8] 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[9], desenvolveram conversores que transformam suas produções em coleções de componentes Java. Este último assume o nome de JavaSketchpad[10].

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].

2.3. Metadados para Objetos de Aprendizado

O LOM [FAR99] se configura como um padrão de metadados dotado de uma estrutura para a descrição de objetos de aprendizado. 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 de um objeto de aprendizado 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 aprendizado, 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 aprendizado, definindo para cada um o seu tipo, domínio e obrigatoriedade.

O IMS Project [WAS99], iniciativa vinculada ao LOM, definiu um conjunto de metadados (IMS metadata) compatíveis com este padrão e implementados na linguagem XML.

2.4. Nova perspectiva a partir dos objetos

Conceber sistemas educacionais como coleções de componentes e organizar atividades pedagógicas que os utilizem como blocos de construção configuram uma metodologia na abordagem do uso das tecnologias na educação, com claras repercussões:

Do ponto de vista do professor / autor do software educacional os componentes, por ele produzidos, podem ser combinados ou recombinados com outros, não se limitando assim a sua produção a um código específico e particular desenvolvido pelo autor (muitas vezes limitado). Além disso, os componentes podem ser facilmente customizados para atender necessidades particulares, promovendo amplamente o intercâmbio de componentes de autores diversos. A Internet facilita e potencializa esse intercâmbio.

Do ponto de vista do aluno, utilizar componentes como blocos de construção promove o desenvolvimento de atividades em que 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. A *Web* cumpre um papel importante de plataforma de comunicação e

---

[8] **Java –** linguagem de programação independente de plataforma criada pela Sun (http://java.sun.com)
[9] **Geometer's Sketchpad** – consiste em um ambiente dinâmico popular de exploração matemática.
[10] **JavaSketchpad** - http://www.keypress.com/sketchpad/java_gsp/

repositório de componentes.

Para que a integração entre os componentes seja possível devem ser estabelecidos padrões para a sua construção, de modo que possam ser livremente intercambiados e combinados. Essencial também é a utilização de padrões para a comunicação entre componentes.

O ESCOT adotou a linguagem Java e os JavaBeans[11] como padrões para a representação dos componentes. Para comunicação foi utilizada a tecnologia InfoBus[12]. Estas soluções impõem restrições aos produtos e sistemas a serem integrados. Seu conjunto de ferramentas deve ser projetado exclusivamente em Java ou conversores que traduzam suas produções para esta linguagem devem ser desenvolvidos.

3. XML, RDF e XSL

As linguagens XML, RDF e XSL, que tem se estabelecido como novos e poderosos padrões na Web, costituem a base de representação na qual o *Anima* está fundamentado.

XML se insere na família das linguagens de marcação. Estas linguagens tem a capacidade de, utilizando marcadores, agregar informações adicionais a documentos com os mais diversos propósitos. No caso de XML os marcadores são caracterizados pelos símbolos "<" e ">". 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.

Tal como HTML, XML é uma linguagem que possui raízes em SGML (Standard Generalized Markup Language). HTML restringe-se a descrever a estrutura do documento do ponto de vista da forma como o mesmo deve ser apresentado ao usuário e não para finalidades de processamento do mesmo. O conjunto de "tags" de HTML é limitado, ou seja, o usuário não pode definir novos marcadores, com suas regras de composição e com a possibilidade de lhes serem atribuídos significados próprios. Já XML, além de linguagem comporta-se também como meta linguagem . XML possui recursos, por exemplo, que permitem a uma comunidade definir seus próprios marcadores, construir, ou estender, linguagens mais apropriadas à representação dos documentos que manipulam.

RDF 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.

Como pode ser observado no exemplo, para descrever um recurso RDF utiliza uma instrução composta de três partes: recurso, tipo de propriedade e valor.

[http://www.framework.org/objeto.xml] --- classe ---> "Botão"

Neste caso, a classe (tipo de propriedade) do objeto definido no documento http://www.framework.org/objeto.xml (recurso) é Botão (valor). Esta instrução escrita em RDF se apresenta como:

```
<RDF:Description about = " http://www.framework.org/objeto.xml ">
   <OBJECT:classe>Botão</OBJECT:classe>
</RDF:Description>
```

[11] **JavaBeans** – padrão de representação de componentes em Java.
[12] **InfoBus** – modelo de comunicação baseado no Java adequado para integração de JavaBeans.

XSL é também uma linguagem descendente da metalinguagem XML. XSL cumpre duas funções: transformar um documento XML em outro documento XML (XSLT) e estabelecer um vocabulário para especificar semântica de formatação.

Para orientar a transformação, um documento XSLT estabelece a forma como cada elemento XML do documento de origem será transformado em outro elemento no documento resultante, que também deve ser um documento XML. Navegadores compatíveis com XSL são capazes de realizar a conversão. É comum a transformação de um documento XML (ou documento escrito em linguagem descendente da metalinguagem XML) em um documento HTML (todo documento HTML pode ter uma versão compatível com a metalinguagem XML).

4. Modelo Integrado de Objetos

O projeto *Anima* se iniciou no contexto de um ambiente para o desenvolvimento de aplicações educacionais denominado Casa Mágica [SAN00]. Casa Mágica oferece suporte para a elaboração de *frameworks* baseadas em objetos e componentes (tal como o exemplo apresentado na figura 1, elaborado neste ambiente) e é capaz de representá-los em XML. Assim, as aplicações podem ser distribuídas pela Internet e executadas diretamente por um módulo *runtime* do sistema ou apresentadas em um navegador *Web*.

O sistema Casa Mágica, no entanto, limita-se a integrar e distribuir objetos desenvolvidos em sua própria linguagem. Por este motivo, procurou-se desenvolver um mecanismo que, além de permitir a integração de *frameworks* produzidos em Casa Mágica com outras linguagens e sistemas baseados em objetos, também pudesse promover a integração de tais linguagens e sistemas entre si. Isto exigiu a concepção de um modelo de objetos suficientemente genérico, denominado *Anima*, para propiciar a integração necessária.

Anima concentra-se em aspectos referentes ao uso de objetos para a construção de aplicações educacionais, mas pode servir de contribuição para um estudo mais abrangente em áreas que se aplica à orientação a objetos, tal como uma representação genérica de objetos na Web, embora este estudo não tenha se preocupado com tal abrangência.

O desafio do modelo definido no projeto *Anima* abrange as seguintes questões:

- **Representação do objeto** – define um formato para representação de objetos, genérico o bastante para ser utilizado por diversas linguagens e sistemas baseados em objetos. Este formato é utilizado para a representação de *frameworks* e para transferência de objetos pela rede.
- **Representação de classe, interface e metadados –** define um formato aberto para representação de informações relacionadas ao objeto, tal como: sua classe, interface, eventos a que reponde, metadados (como os definidos pelo LOM), etc.
- **Mecanismo de conversão –** estabelece um mecanismo para converter *frameworks* de sua representação genérica para uma representação específica em uma ou mais linguagens ou sistemas baseados em objetos.
- **Comunicação entre objetos –** define um mecanismo padronizado de comunicação entre objetos que compõem uma *framework*.

Neste trabalho iremos abordar as três primeiras questões no Sistema *Anima*.

Toda a estrutura do *Anima* está baseada em um modelo de objetos que, como foi abordado em tópicos anteriores, tem a capacidade de representar agentes e componentes.

4.1. Representação do objeto

Em sua concepção original, os objetos são categorizados sob a forma de classes. Classes são abstrações genéricas que estabelecem a estrutura de atributos e comportamento que caracterizam cada objeto a ela pertencente. Elas não definem, no entanto, o estado em que se encontra cada atributo do objeto.

Como podemos observar na figura 2, se tomarmos dois objetos pertencentes à mesma classe eles irão se diferenciar a partir do estado distinto em que se encontram, estado este caracterizado pelos valores de seus atributos.

Componente	Propriedades
Liga	**Rótulo:** Liga **Largura:** 60 **Altura:** 40

Componente	Propriedades
Desliga	**Rótulo:** Desliga **Largura:** 50 **Altura:** 20

Figura 2: Componentes com variação nas propriedades

Assumindo que temos uma estrutura e comportamento já estabelecidos para a classe, o que caracterizará cada objeto individual da mesma classe será sua identidade e valores de estado de seus atributos.

Se dois computadores compartilham a mesma biblioteca de classes, uma *framework*, como a ilustrada na figura 1, pode ser transformada em um arquivo que conterá a codificação das identidades dos componentes e valores de suas propriedades - processo conhecido como serialização. O arquivo serializado pode ser transmitido para outra máquina que poderá utilizá-lo para reconstruir a *framework*.

O sistema Casa Mágica utiliza a linguagem XML para a serialização de seus objetos. Abaixo está ilustrada a serialização do botão da figura 2.

```
<object id=”botao1” class="Botao">
  <attribute name="rotulo">Liga</property>
  <attribute name="largura">60</property>
  <attribute name="altura">40</property>
</object>
```

Por ser um formato que pode ser igualmente aplicado a outras linguagens e sistemas baseados em objetos, o mesmo padrão foi adotado na representação de objetos em *Anima*.

4.2. Representação da classe, interface e metadados

Em muitas linguagens de programação orientadas ao objeto, a ligação entre o comportamento de um objeto (definido por métodos) e o seu estado, é feita sob a forma de um ponteiro de memória que liga as variáveis de estado (atributos do objeto) aos métodos (reunidos em uma classe). Deste modo: “*A classe (em particular, os métodos que ela define) define o caminho pelo qual o estado poderá (e será) interpretado no sistema, e portanto é uma forma de metadado para o estado. Como resultado, a ligação entre um objeto e sua classe é essencialmente uma ligação de metadado*”. [MAN98]

No modelo *Anima* os valores dos atributos se encontram em um documento XML, e os métodos que definem seu comportamento estão em arquivos com codificação específica da linguagem utilizada para implementa-los. Um documento RDF é utilizado para estabelecer a ligação entre os atributos e os métodos. RDF é utilizado ainda para agregar metadados, segundo a especificação do IMS, às classes dos objetos e aos objetos.

4.3. Ferramentas e processo de construção

Como é ilustrado na figura 3, o processo de produção de uma aplicação, utilizando o modelo *Anima* inicia-se a partir de um conjunto de classes escritas em uma linguagem específica. A partir delas, uma ferramenta gera automaticamente um arquivo RDF contendo a descrição de suas interfaces.

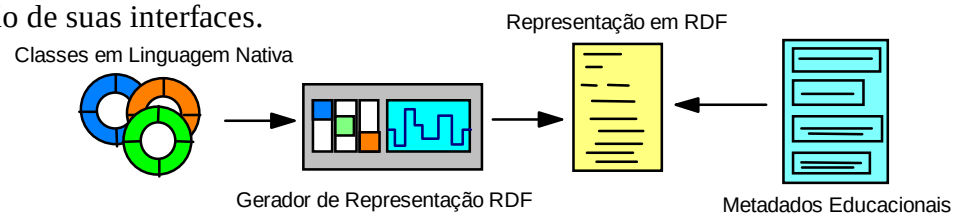

Figura 3: Montagem da Representação Genérica

A partir dos dados em RDF outra ferramenta de produção é utilizada na construção da *framework*. Neste caso, a ferramenta pode ser operada em qualquer ambiente de produção, em qualquer linguagem ou plataforma capacitada a ler um arquivo RDF. No projeto *Anima* o ambiente de produção do Casa Mágica foi adaptado para este fim.

Figura 4: Montagem da *framework* de Objetos utilizando a Representação RDF

Durante o processo de produção da *framework* (figura 4), no momento em que o usuário seleciona um determinado componente, a ferramenta decide se está habilitada a instanciar um objeto na linguagem em que a respectiva classe foi escrita. Isto depende da compatibilidade entre a plataforma da ferramenta e a da classe. Ambas podem estar na mesma plataforma, em plataformas compatíveis ou em plataformas que possuam interfaces que lhes permita trabalhar em conjunto.

Se isto for possível, o usuário poderá, durante o processo de produção, manipular diretamente o componente, configurá-lo e experimentar seu comportamento. Caso contrário, a ferramenta ainda tem acesso, a partir do arquivo RDF, a um conjunto suficiente de informações sobre a interface e apresentação do objeto, que possibilitam sua inserção e integração na aplicação, mesmo que ele não possa ser experimentado interativamente.

Embora a ferramenta de produção não precise necessariamente estar habilitada a lidar com as especificidades de linguagem de todas as classes, é necessário que a plataforma de execução o consiga.

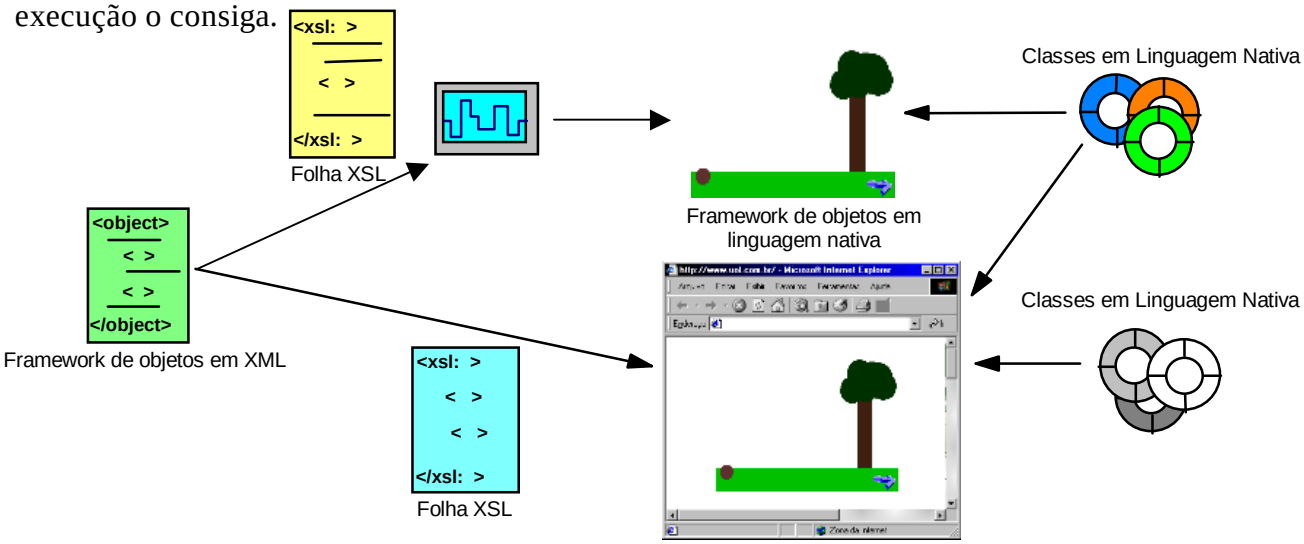

Figura 5: Processo de execução de uma aplicação

A *framework* produzida em XML pode ter destinos diversos: ser transformada em uma *framework* em linguagem nativa ou se transformar em um documento apropriado para ser executado em um navegador Web. Como pode ser observado na figura 5, a mesma *framework* pode se converter em dois formatos diferentes com a variação da folha XSL.

Se as classes de todos os objetos da *framework* forem implementadas na mesma plataforma, tal como Java, esses objetos podem ser convertidos de XML para a plataforma específica, utilizando-se para tal as facilidades de folhas de estilo XSL. Um módulo adicional

executa a compilação do código fonte resultante para o código compilado ou semicompilado.

Para composições de *frameworks* com objetos em linguagens diferentes, foi escolhido como ambiente integrador o navegador Web, dadas as múltiplas linguagens que este suporta ou pode vir a suportar a partir de Plugins. Para isto, a folha XSL é configurada para converter a *framework* diretamente em um documento HTML com a inserção dos múltiplos objetos que a compõe. Esta conversão é executada pelo próprio navegador, se o mesmo suportar XML e XSL, ou pelo servidor, em caso contrário.

A inserção dos objetos no documento HTML varia de acordo com a linguagem No caso de Java, por exemplo, a inserção é realizada através da inclusão de Applets.

5. Considerações Finais

O trabalho apresentado relata uma parte de um projeto mais amplo que se estende a outras questões, como o tratamento de eventos e a comunicação entre objetos, principalmente quando estes são implementados em linguagens diferentes.

As experiências realizadas com o *Anima*, baseadas no protótipo implementado dentro do ambiente do sistema Casa Mágica, tem comprovado que a adoção deste modelo mais genérico, independente de uma implementação em linguagem específica, possui um forte poder de integração e amplia o universo de representação dos objetos, tornando natural sua integração com outras estruturas, como metadados, por exemplo.

Referências Bibliográficas

[ADL00] **Adler , Sharon et al.**, Extensible Stylesheet Language (XSL) Version 1.0, W3C Working Draft 27 March 2000, [OnLine] http://www.w3.org/TR/xsl/.

[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.

[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.

[LAS99] **Lassila, O., Swick, R. 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/.

[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.

[MEY88] **Meyer, B.** Object - Oriented Software Construction. USA: Prentice Hall International Ltd., 1988.

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

[REP96] **Repenning, A. Ambach, J.** 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.

[ROS98] **Roschelle, J. Digiano, C. Pea, R. Kaput, J.** Educational Software Components of Tomorrow (ESCOT). SRI International, 1998 [Online] http://www.escot.org/escot/External/MSET_ESCOT.html.

[RSC98] **Roschelle, J., Kaput, J., 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.

[SAN00] **Santanchè, André**. 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.

[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.

[W3C98] **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.

[WAS99] **Wason, Thomas D.** IMS Meta-Data - Specification: Draft, February 1999. IMS Project of EDUCAUSE, [Online] http://www.imsproject.org/work_public/meta-data_did188.html.