Adaptação do método OOHDM para publicação de aplicações hipermídia em Adobe Flex

Pedro Germani Ghiorzi[1], Patrícia Vilain[1], Daniel Schwabe[2]

[1] INE – UFSC / [2] INF – PUC-Rio
Brasil
pedro.germani@gmail.com, vilain@inf.ufsc.br, dschwabe@inf.puc-rio.br

Abstract. The utilization of Flex in the development of hypermedia applications has increased significantly lately. However, development is usually done in an ad hoc manner, due to the lack of specific methods. Aiming tor proposes an adaptation of the OOHDM method so it can be applied to Flex development as a way to overcome this deficiency. We suggest four extensions to be incorporated to the Abstract Interface Design and Implementation phases of OOHDM. The first one maps the attributes of navigational classes and entities from UIDs (User Interaction Diagrams) to elements of the abstract interface. The second extension classifies each MXML component to a class of in the Abstract Widgets Ontology. The third extension presents a new syimbology to represent widgets in an ADV (Abstract Data iew) and the fourth one introduces a mapping from ADVs to the View States feature of the Adobe Flex IDE.

Keywords: Hypermedia, OOHDM, Interface Design, Adobe Flex, Adobe Flash, *view states*, MXML Components.

1 Introdução

As aplicações de internet se tornam cada dia mais avançadas e complexas, aumentando seu alcance na combinação de elementos multimídia e incorporando novas tecnologias rapidamente.

O Adobe Flex[1] é um ambiente de desenvolvimento de aplicações ricas de internet (*rich internet applications* - RIAs) que gera aplicações que são executadas no Adobe Flash Player, hoje em dia um dos softwares com maior penetração nos computadores ligados à internet [2]. Comcriação da linguagem orientada a objetos *ActionScript 3.0,* as "aplicações Flash"[3] deixam de ser apenas elementos componentes de um sistema HTML, por exemplo, e passam a ser soluções para a criação da aplicação como um

1 http://www.adobe.com/products/flex/overview
2 http://www.adobe.com/products/player_census/flashplayer/

todo. Apesar de se tornarem mais completas e de serem largamente utilizadas o que se vê é que atualmente a grande maioria das aplicações publicadas na plataforma Flash é feita de maneira *ad hoc* e muitas vezes sem planejamento adequado para prover recursos como reusabilidade e manutenção. A grande maioria dos métodos de desenvolvimento encontrados pelos autores tem foco não em metodologias genéricas, ou de desenvolvimento de sistemas hipermídia, mas sim aplicações específicas, como *XML handling*.

Com o intuito de melhorar o desenvolvimento em Adobe Flex de sistemas hipermídia, há a necessidade de um método de modelagem que incorpore, além dos aspectos normalmente modelados pelos métodos tradicionais, questões relativas à navegação e também à criação de interfaces complexas e personalizadas. N. Koch [2] comenta que normalmente os métodos tradicionais de modelagem não consideram o projeto de navegação como um processo separado do projeto de interfaces com o usuário.

Dentre diversoso métodos propostos na literatura, um dos mais maduros é o *Object Oriented Hypermedia Design Method* (OOHDM) [6]. Este método apresenta todas as etapas necessárias para o projeto de um sistema hipermídia permitindo, inclusive, o projeto de interfaces com o usuário de maneira abstrata.

Baseando-se na experiência dos autores no uso do OOHDM, este trabalho tem como objetivo adaptar o método OOHDM para o desenvolvimento de sistemas hipermídia em Adobe Flex, permitindo também uma flexibilidade na incorporação de novos componentes de programação na medida em que o desenvolvedor utiliza o método.

O restante do artigo está organizado da seguinte maneira: as seções 2 e 3 apresentam, respectivamente, o método OOHDM e o ambiente de desenvolvimento do Adobe Flex. Na seqüência, a seção 4 apresenta as extensões propostas ao método OOHDM. E, por fim, na seção 5 são apresentadas as conclusões.

2 OOHDM

O método *Object-Oriented Hypermedia Design Method* (OOHDM) [5] [6] é um método de desenvolvimento de sistemas hipermídia. Ele utiliza a noção de que os objetos de navegação são visões dos objetos conceituais; o uso de abstrações apropriadas para organizar o espaço navegacional, com a introdução do conceito de contexto navegacional; e a separação de questões de interface e questões de navegação.

O OOHDM é um método iterativo composto de 5 etapas principais: Análise de Requisitos, Modelagem Conceitual, Projeto de Navegação, Projeto de Interface Abstrata e Implementação. A seguir as três primeiras etapas são apresentadas resumidamente e a quarta etapa é detalhada com maior profundidade porque é o foco deste trabalho.

---

3 Como serão denominadas as aplicações que executam no *Flash Player* criadas no Adobe Flex ou no Flash IDE.

2.1 Análise de requisitos

O primeiro passo para a construção de um sistema, de acordo com o método OOHDM é a análise e aquisição dos requisitos que irão determinar as funcionalidades e comportamentos do sistema a ser projetado. Entre os artefatos gerados estão os casos de uso e os diagramas de interação do usuário (UIDs) [7]. Os UIDs capturam a troca de informações entre o usuário e a aplicação nos casos de uso identificados.

2.2 Modelagem Conceitual

Durante esta etapa do processo, é construído um modelo do domínio da aplicação, resultando num diagrama de classes conceituais similar ao propost na *Unified Modeling Language (UML).*

2.3 Projeto de Navegação

Nesta etapa do OOHDM é descrita a estrutura de navegação da aplicação hipermídia que representa as visões dos objetos conceituais que o usuário terá acesso e as conexões entre estes objetos. Esta estrutura é organizada em termos de contextos navegacionais, que são conjuntos de objetos de navegacão que são obtidos a partir de classes de navegação, como chamado nodos. Além dos nodos, são também definidas estruturas de acesso tais como âncoras, índices e guias

A modelagem navegacional gera dois importantes artefatos: o diagrama de classes navegacionais e o diagrama de contextos navegacionais. O primeiro define todos os objetos navegáveis do sistema de acordo com visões sobre os objetos conceituais, e o segundo as possíveis navegações entre estes objetos navegáveis.

2.4 Projeto de Interface Abstrata

A modelagem de uma interface abstrata é construída definindo objetos perceptíveis em termos de interface e definindo como se dá a comunicação entre o usuário e objetos navegacionais através da interface. Estes objetos são mapeados a partir dos objetos navegacionais do projeto de navegação, provendo assim uma aparência perceptível do sistema. O OOHDM utiliza as seguintes técnicas durante o projeto da interface abstrata: Ontologia de Widgets Abstratos, Ontologia de Widgets Concretos e Abstracts Data Views (ADVs).

2.5 Ontologia de *Widgets* Abstratos

Os *widgets* são elementos de interface com o qual o usuário pode interagir. Neste caso eles são abstratos, ou seja, são descritos modelos de interface especificando como os objetos navegacionais serão apresentados e quais elementos serão

perceptíveis para o usuário, sem expressar sua forma ou funcionamento. Posteriormente estes modelos são mapeados para uma ontologia que descreve os *widgets* concretos.

As classes dessa ontologia representam os tipos de elementos abstratos das interfaces das aplicações hipermídia. Estas classes focam no papel que o widget pode cumprir na interface – exibir valores, capturar valores ou sinalizar eventos.

2.6 Ontologia de Widgets Concretos

Utilizando-se dos elementos e primitivas da Ontologia de Widgets Abstratos, o processo de criação da interface modela e mapeia os elementos de interface desejados para a ontologia abstrata. A partir das descrições dos elementos abstratos, é gerada outra descrição, desta vez mapeando cada elemento abstrato para um elemento concreto, que possui descrições mais próximas das interfaces reais, considerando de maneira ainda rasa questões relativas ao ambiente em que será executada a aplicação.

3 ADOBE FLEX

O "Adobe Flex é um framework *open-source* para criação de aplicações altamente interativas, que são publicadas na maioria dos browsers através do Flash Player ou até mesmo no desktop (em diferentes sistemas operacionais: Linux, MacOS e Windows) utilizando o ambiente de execução Adobe AIR[4].

Com Adobe Flex, podemos elaborar aplicações para web de maneira similar ao código HTML, usando-se uma nova linguagem de programação no paradigma de linguagens de marcação, baseada em XML, chamada de MXML [1], que possibilitou essa analogia.

A MXML está é baseada em classes Actionscript 3.0 [3], ou seja, utiliza-se da mesma estrutura do Flash, mas com a possibilidade de o programa ser escrito de outra forma. No tempo de compilação, as marcações MXML são transformadas em instâncias das classes ActionScript 3.0, isto é, o programa pode ser escrito utilizando de maneira híbrida MXML e ActionScript. Normalmente, a primeira faz a composição dos elementos de interface, suas propriedades e eventos associados, enquanto a segunda descreve a lógica da aplicação.

4 Estendendo o OOHDM

A fim de aperfeiçoar a tarefa de mapeamento de um modelo abstrato de interface OOHDM para implementação da interface concreta em um sistema baseado em Flex,

4 <http://www.adobe.com/products/flex/overview/>

este trabalho sugere algumas adaptações às etapas de Projeto de Interface Abstrata e Implementação do OOHDM.

A Fig. 1 mostra as etapas do OOHDM e os passos de cada etapa. Os artefatos gerados e que são trocados entre as etapas estão representados junto às transições. As extensões propostas no presente trabalho estão em negrito.

São propostas quatro extensões ao processo OOHDM. Estas extensões são detalhadas a seguir. Os exemplos utilizados são parte do estudo de caso e foram extraídos do projeto da tarefa de pesquisa de uma rádio.

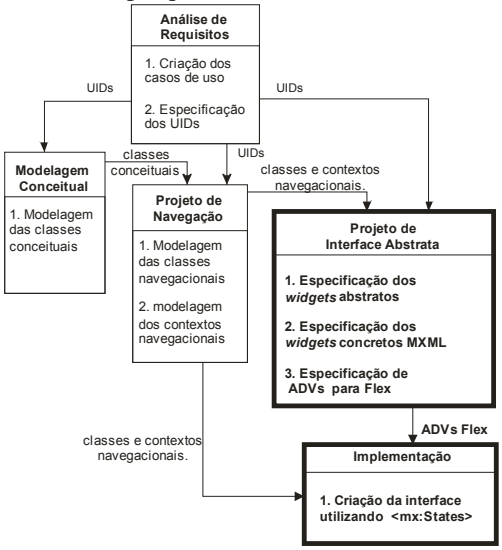

Fig. 1. Esquema do processo OOHDM adaptado para criação de sistemas Flex.

Extensão 1 - Especificação dos *widgets* abstratos

Esta extensão ao processo OOHDM utiliza uma proposta sugerida por [4] para mapear os atributos das classes navegacionais e entidades dos UIDs para os elementos da interface abstrata. Este mapeamento é utilizado na construção da interface abstrata para indicar a qual classe da Ontologia de *Widgets* Abstratos com a qual cada atributo e operação das classes navegacionais pode ser relacionado, de acordo com seu "papel" nos UIDs.

A tabela 1 apresenta as regras de mapeamento, indicando a classe da Ontologia de *Widgets* Abstratos à qual cada tipo de elemento do UID está associado.

Tabela 1. Mapeamento de elementos dos UIDs para ontologia de widgets abstratos [4].

UIDs	***Widgets* Abstratos**
Item de dado e Item de dado personalizado	*ElementExhibitor* ou *SimpleActivator*

Estrutura	A estrutura toda pode ser mapeada para um *ElementExhibitor* ou *SimpleActivator*, assim como cada elemento da estrutura pode ser mapeado para um *ElementExhibitor* ou *SimpleActivator*. No segundo caso, a Estrutura deve ser mapeada para um *CompositeInterfaceElement.*
Conjunto, Conjunto com conteúdo personalizado e Conjunto em disposição diferenciada	1º passo: mapear o conjunto para um *CompositeInterfaceElement*. 2º passo: mapear cada elemento do conjunto para um *ElementExhibitor* ou *SimpleActivator*.
Dado opcional	*ElementExhibitor* ou *VariableCapturer*. No caso de *VariableCapturer*, o elemento pode ser mapeado para: *IndefiniteVariable*, *ContinuousGroup*, *DiscreetGroup, MultipleChoices* ou *SingleChoices*.
Entrada do Usuário	*IndefiniteVariable*
Entrada do Usuário Enumerada	*SingleChoice, MultipleChoice*
Saída do Sistema	Recebe o mapeamento descrito para os elementos dos UIDs, conforme o tipo de elemento que representa a Saída do Sistema (Item de dado, Item de dado personalizado, etc.).
Texto	*ElementExhibitor*
Estado de Interação, Estado Inicial da Interação e Estados Alternativos da Interação.	*CompositeInterfaceElement*. Caso o Estado de Interação seja um conjunto de *CompositeInterfaceElement*, o estado de interação será a associação dos *CompositeInterfaceElements*.
Sub-estados de um Estado de Interação.	*CompositeInterfaceElement*
Transição com Seleção da Opção X e Transição com Seleção da Opção Restrita X.:	*SimpleActivator*. No caso da opção precisar de outro elemento para que o estado de interação destino se torne o foco da interação, a mesma deve ser mapeada para *MulipleChoice* ou *SingleChoice* e atrelada a um *SimpleActivator*.
Transição com Seleção de N Elementos	*ContinuousGroup*,DiscreetGroup*, MultipleChoices* ou *SingleChoices*.
Chamada de Outro UID, Chamada a partir de Outro UID, Transição com Condição Y, Pré-Condições, Pós-Condições e Notas Textuais.	Não há mapeamento para a ontologia de *widgets* abstratos.

A extensão 1 propõe a construção de uma tabela relacionando os atributos das classes navegacionais aos elementos dos UIDs e, então, aos *widgets* abstratos. Também são considerados os elementos dos UIDs que não são mapeados para com atributos das ces navegacionais.

A Fig. 3 apresenta parte de um diagrama de classes navegacionais cujas classes navegacionais estão relacionadas com as informações que aparecem no UID Pesquisar Rádio (Fig. 2)

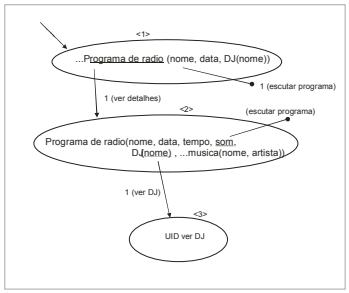

Fig 2. UID Pesquisar Rádio

As três primeiras colunas da tabela 2 mostram os *widgets* abstratos oriundos do UID e do esquema de classes navegacionais definido na Fig. 3.

Com este mapeamento, obtém-se o primeiro conjunto de informações para o projeto da interface abstrata identificando que tipo de exibição terá cada atributo da classe navegacional.

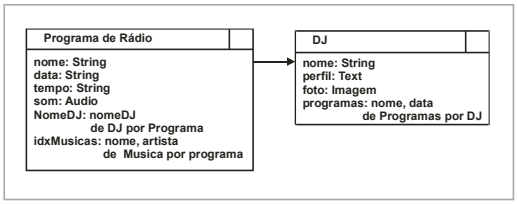

Fig. 3. Diagrama de classes navegacionais

Extensão 2 - Especificação dos *widgets* concretos MXML

A partir de uma comparação entre as premissas da Ontologia de *Widgets* Abstratos e os componentes de interface MXML, foi definida uma nova Ontologia de *Widgets* Concretos, onde cada componente MXML é classificado de acordo com as classes da Ontologia de Widgets Abstratos. A Fig. 4 apresenta esta ontologia

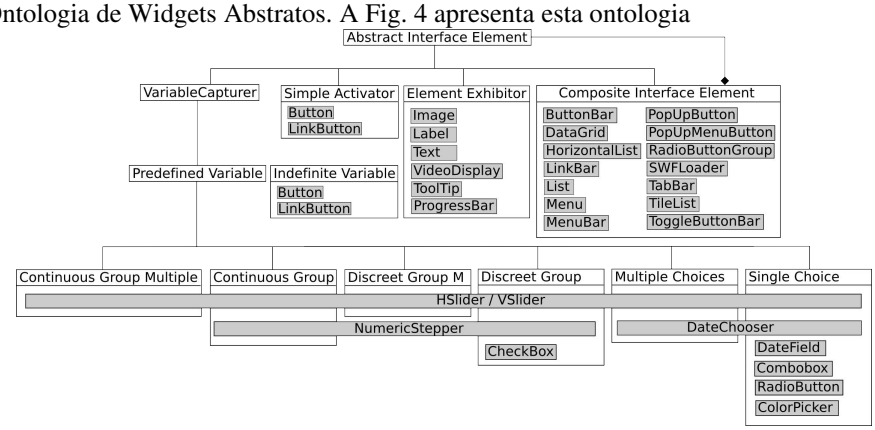

Fig. 4. Sobreposição das classes de widgets abstratos

Com esta nova ontologia a definição de *widgets* concretos MXML pode ser determinada e a coluna com a informação de qual *widget* concreto deve ser utilizado é definida na tabela 2

Tabela 2. Exemplo de definição de widgets abstratos e concretos

Classe: Programa de Rádio			
Atributos	**origem nos UIDs**	***Widgets* Abstratos**	**Widgets Concretos**
nome:String	Item de dado	*ElementExhibitor*	*mx:Label*
data:String	Item de dado	*ElementExhibitor*	*mx:Label*
tempo:String	Item de dado	*ElementExhibitor*	*mx:Label*
Som:Audio	Item de dado transição(escutar programa)	*SimpleActivator*	*mx:Button*

nomeDJ:String	Item de dado transição (ver DJ)	*SimpleActivator*	*mx:LinkButton*
idxMusicas	Estrutura	*CompositeInterfaceElement*	*mx:VBox*

Extensão 3 – Especificação de ADVs para Flex

Para realizar a construção dos ADVs incluindo o resultado da aplicação dos mapeamentos anteriores, este trabalho apresenta uma simbologia para a representação dos *widgets* nos ADVs. Cada ADV, ADV aninhado ou atributo é acompanhado de um símbolo que corresponde à classe da ontologia de *widgets* abstratos e do nome do widget concreto ao qual foi mapeado. Os ADVs que incluem esta simbologia são chamados de "ADVs Flex". Optou-se por utilizar uma notação própria em vez de utilizar estereótipos da UML porque estes não fazem diferenciação gráfica entre os diversos tipos de elementos, e portanto dificultam a compreensão visual da semântica do modelo sendo representado.

.

A Fig. 5 mostra a legenda correspondente às classes abstratas e um exemplo de um ADV Flex, ou seja, um ADV incluindo a notação estendida.

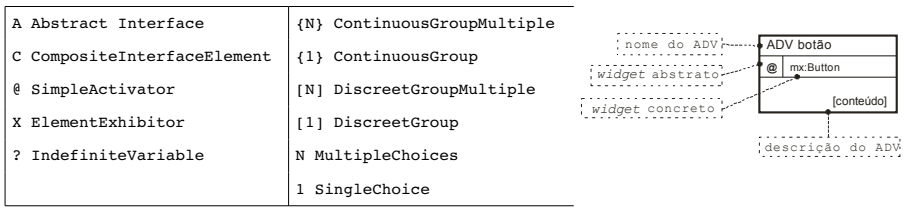

Fig. 5. Legenda da representação da classe de widgets abstratos nos ADVs e exemplo de ADV Flex.

Com a reunião das informações geradas a partir dos diagramas de classes e contextos navegacionais e da aplicação das extensões 1 e 2 propostas neste trabalho, s são especificado ADVs para cada nodo e atributo do esquema de classes navegacionais e para cada índice e contexto do diagrama de contextos navegacionais. Os ADVs que representam os nodos (classes) navegacionais são chamados de "ADVs Primários", ou "ADVs Base". Os ADVs relacionados a contextos das classes navegacionais são chamados de "ADVs Secundários". Os ADVs relacionados a atributos são chamados de "ADVs Opcionais". Os índices do diagrama de contextos navegacionais devem ser descritos por "ADVs Especiais", que farão parte do contexto da aplicação e permitirão a navegação para os objetos navegacionais. Esta hierarquia foi criada para acomodar os ADVs Flex em *view states* do Adobe Flex (representados pela marcação <mx:states>) no momento da implementação do sistema.

Extensão 4 - Especificação de *view states* do Flex

Esta extensão propõe a definição de *view states* do Flex a partir dos ADVs Flex. Como os *view states* são organizados de maneira hierárquica, a organização dos ADVs Flex em primários, secundários e opcionais nos permite um mapeamento direto para os *view states* do Flex.

A seguir é mostrado um exemplo de definição dos estado de interface Flex de acordo com a extensão 4. A partir dos ADVs Flex, a interface correspondente a eles pode começar a ser construída. Implementa-se primeiro o ADV Primário correspondente a um nodo navegacional, de maneira que ele seja o *base state* do componente MXML escolhido para representá-lo. No exemplo foi escolhido o componente mx:Panel, portanto a definição deste componente com os elementos internos do ADV correspondente é implementado no *base state* de acordo com o mapeamento.

```
 <mx:Panel
xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:Label id="nome_lbl" text="Nome" />
    <mx:LinkButton id="dj_lbtn" label="DJ"/>
    <mx:Button id="som_btn" label="Som"/>
   <mx:Panel>
```

Em seguida são definidos os *view states* que mapeiam cada ADV Secundário para um mx:state:

```
   <mx:Panel
xmlns:mx="http://www.adobe.com/2006/mxml">
    <!-- Declaração da variável states,
   que indica que este mx:Panel terá mais de um
estado -->
    <mx:states>
     <!--Declaração do primeiro estado: Programa de
Rádio por Data-->
      <mx:State name="porData">
       […]
      </mx:State> [...]
```

Fig 6. Interface para o ADV Programa de Rádio por Data (à direita)

Com a interface definida são determinados os valores dos atributos dos componentes baseados nos atributos dos objetos navegacionais os quais devem ser implementados naturalmente como classes ActionScript.

Para validar a proposta deste trabalho foi realizado um estudo de caso que trata de uma estação de rádio *on-line* na qual o usuário também pode fazer consultas aos programas exibidos anteriormente na rádio. O sistema foi implementado utilizando o modelo MVC. As classes do modelo navegacional foram traduzidas em classes Actionscript, e os ADVs Flex para MXMLs ou *view states* destes. O controle do sistema foi construído de maneira híbrida, com scripts que disparam a instanciação do programa principal embutidos em tags *<mx:script>*. A Fig. 6 mostra uma tela do estudo de caso.

5 CONSIDERAÇÕES FINAIS

Neste trabalho o método OOHDM foi estendido para ser utilizado na publicação de aplicações construídas no ambiente Flex. A utilização das extensões propostas faz com que o projeto da interface com o usuário seja feito de acordo com as capacidades e características dos componentes MXML.

A partir das ontologias de *widgets de interface* foi possível criar elementos de projeto que funcionam como estruturas para realizar o mapeamento direto do projeto de interface abstrata para a implementação do sistema, extraindo dos ADVs Flex informações para a construção do código MXML que compõe parte da estrutura de uma aplicação Flex.

As descrições providas pelo método OOHDM, a construção de sistemas hipermídia em Flex se torna organizada e claramente descrita. Na aplicação gerada,. O Adobe Flex implementa a interface com o usuário, o controle da lógica do sistema, o acesso ao sistema de informações, as requisições HTTP; enquanto o Flash gera gráficos e componentes de programação inovadores no que diz respeito à usabilidade e apresentação visual. Com as descrições providas pelo método OOHDM, a construção de sistemas hipermídia em Flex se torna organizada e claramente descrita.

Apesar deste trabalho ter seu enfoque em aplicações feitas em Flex, as extensões sugeridas (principalmente as extensões 1, 2 e 3) podem ser aplicadas em outros ambientes de programação. Neste caso, os elementos descritos nos ADVs Flex podem conter os componentes ou objetos da outra linguagem utilizada (por exemplo, OpenLaszlo[5]), em vez de componentes MXML.

6 REFERÊNCIAS

1. COENRAETS, C. An overview of MXML: The Flex markup language. Adobe - Developer Center. 2003. disponível em <http://www.adobe.com/devnet/flex/articles/paradigm.html> Acesso em: 03 Abr. 2008.
2. KOCH, N. A Comparative Study of Methods for Hypermedia Development. Universidade Ludwig-Maximilians de Munique, Alemanha, 1999.
3. MOOCK, C. Essencial ActionScript 3.0. Sebastopol, CA, EUA:O'Reilly, 2007.
4. REMÁCULO, L. R. Personalização de Diagramas de Interação do Usuário e Mapeamento para a Ontologia de Widgets Abstratos, Trabalho de conclusão de curso, UFSC, Brasil, 2005
5. ROSSI, G. ; SCHWABE, D. . Modeling and Implementing Web Applications with OOHDM. In: Rossi, G., Pastor, O., Schwabe, D.,

Olsina, L.. (Org.). Web Engineering: Modelling and Implementing Web Applications. 1 ed. New York, London, Heidelberg: Springer Verlag, 2007, v. 1, p. 109-159..

6. SCHWABE, D.; ROSSI, G. V. . An Object Oriented Approach To Web-Based Application Design. Theory and Practice of Object Systems, New York, v. 4, n. 4, p. 207-225, 1998.
7. VILAIN, P.; SCHWABE, D.; de SOUZA, C.S. A Diagrammatic Tool for Representing User Interaction. In: Proceedings of the UML 2000 Conference. Springer-Verlag, Berlin: (2000) 133-147

5 http://www.openlaszlo.org/