Geração de Aplicações Hipermídia para Televisão Digital Baseada em Desenvolvimento Orientado a Modelos

Tales P. Nogueira[1], Cidcley T. de Souza[2], Mariela I. Cortés[1]

[1] Mestrado Acadêmico em Ciência da Computação (MACC)
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 – Campus do Itaperi – Fortaleza – Ceará – Brasil

[2] Núcleo Avançado em Engenharia de Software Distribuído e Sistemas Hipermídia (NASH)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Av. Treze de Maio, 2081 – Benfica – Fortaleza – Ceará – Brasil
{tales.paiva, mariela}@larces.uece.br, cidcley@cefetce.br

Abstract. A hypermedia application can be defined as a system composed by various kinds of information, like text, data, graphics, video and audio, connected by a hypertext program. Besides the Internet, one of the most appropriate vehicles that can be use to run hypermedia applications is the digital television. However, there are still few alternatives to the generation of this type of content. This paper aims to describe how model driven architecture can help the authoring of hypermedia applications which can be interactive or not. In order to achieve this, a UML profile, called UML-TV is proposed and some usage examples are shown.

Keywords: digital television, UML profile, model driven development

1. Introdução

No sistema de TV digital adotado pelo Brasil, ISDB-TB (*Integrated Services Digital Broadcasting – Terrestrial*) [7], o *middleware*[1], denominado Ginga, é composto por duas partes: Ginga-NCL [12] e Ginga-J [15] que são especializados em processar, respectivamente, aplicações declarativas em NCL e aplicações procedurais em Java.

Para criar uma aplicação declarativa que possa ser executada no Ginga-NCL, por exemplo, o desenvolvedor, atualmente, tem duas opções. A primeira é criar o documento manualmente através de um editor de textos qualquer ou com a ajuda de um *plug-in* para o IDE (*Integrated Development Environment*) Eclipse chamado NCL Eclipse [2]. A outra opção para se criar um documento NCL é utilizar a ferramenta Composer [8], a qual oferece diversas perspectivas (textual, estrutural, temporal e *layout*) de visualização do documento criado.

Nesse contexto, pode-se inferir que novas ferramentas que auxiliem a criação de aplicações interativas podem ser propostas de forma que novos paradigmas sejam utilizados na tentativa de melhorar o processo de produção dessas aplicações.

[1] Camada intermediária que possibilita a interoperabilidade entre o hardware e o software.

Um paradigma que pode ser utilizado para esse fim é o da Arquitetura Orientada a Modelos (MDA – do inglês *Model Driven Architecture*) [14], cuja principal característica é tornar os modelos, e.g. um diagrama de classes UML, um artefato de primeira classe no ciclo de desenvolvimento de software. Este paradigma é baseado em transformações de modelos independentes de plataforma em modelos específicos de plataformas onde, finalmente, os modelos intermediários podem ser transformados em código fonte.

Uma das formas de se aplicar os conceitos de MDA é através da especificação de perfis UML, que permite adaptar a UML a determinado domínio por meio da criação de estereótipos, atributos e restrições, criando, assim, uma linguagem de domínio específica (DSL – do inglês *Domain Specific Language*).

Este trabalho tem como objetivo utilizar conceitos de MDA para a produção de aplicações interativas através da especificação de um perfil UML e as devidas transformações entre modelos. A definição de uma linguagem de domínio especifico possibilita a modelagem de componentes e operações próprias do domínio, tais como as operações de sincronismo entre mídias. Além disso, possibilita a geração do código da aplicação interativa correspondente com o intuito de facilitar e tornar mais rápida a especificação e produção de aplicações hipermídia.

Este artigo está organizado da seguinte forma: na Seção 2, são citados alguns trabalhos relacionados; na Seção 3, o perfil UML-TV é especificado; na Seção 4, exemplos de utilização da proposta são demonstrados. Finalmente, a Seção 5 contém as considerações finais e os trabalhos futuros.

2. Trabalhos Relacionados

Vários trabalhos que propõem DSLs através de perfis UML com o objetivo de adaptar diagramas UML a domínios específicos podem ser citados [1, 6, 9, 16]. Ziadi, Helouet e Jezequel [16], por exemplo, apresenta um perfil UML para linhas de produto de software que especifica a variabilidade de produtos por meio de diagramas de classe e de sequência.

Sauer e Engels [11] propõem uma extensão da UML para a especificação integrada de sistemas multimídia baseada em um método de desenvolvimento orientado a objetos. Para isso, o comportamento do sistema é modelado através de diagramas de máquina de estados para especificar comportamento interativo, e de diagramas de sequência para especificar o comportamento ao longo do tempo. No entanto, os autores não contemplam nenhum tipo de interatividade do sistema com o usuário.

O trabalho de Bertino e Ferrari [4] apresenta uma ferramenta para especificação e geração de aplicações multimídia. Embora os tipos de apresentação tratados não incluírem aplicações televisivas, as operações de sincronismo entre mídias são bem exploradas. Como ponto fraco identificado nesse trabalho, está a falta de uma interface gráfica mais intuitiva que utilize elementos visuais que representem as mídias. Como proposta deste artigo para solucionar esse problema, introduzimos a utilização da notação UML para representar as mídias e seus relacionamentos.

Um perfil UML é apresentado em [3] com o objetivo de modelar aspectos de navegação e apresentação de cenários hipermídia. Os autores empregaram os

diagramas de estado e de atividades para alcançar seus objetivos. Contudo, diferentemente do que é proposto neste artigo, o comportamento interativo do usuário não é modelado, e sim apenas as relações entre as mídias.

3. Perfil UML-TV

O processo proposto neste trabalho tem como objetivo prover uma alternativa melhor para os desenvolvedores de aplicações interativas para TV digital especificarem e gerarem o conteúdo hipermídia em qualquer padrão adotado. Como pode ser visto na Fig. 1, o perfil UML (UML-TV) é utilizado na modelagem das aplicações. Posteriormente o modelo passa por transformações que geram código específico de linguagens declarativas ou imperativas, dependendo das transformações aplicadas.

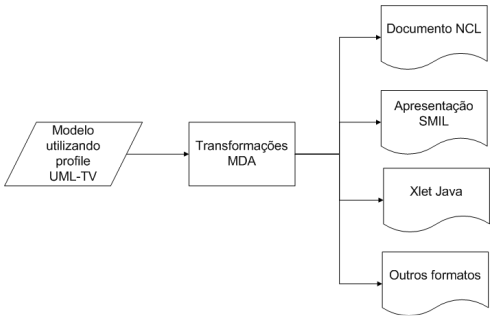

Fig. 1. Visão geral da proposta

A versão atual da UML define alguns tipos de diagramas dinâmicos que podem ser utilizados para a modelagem de aspectos de sincronização temporal em aplicações interativas para TV. Neste trabalho, o Diagrama de Comunicação, introduzido na versão mais recente da UML, é utilizado para ilustrar a colaboração entre objetos e as trocas de mensagens ao longo da execução da aplicação.

Diagramas de comunicação representam uma combinação de informações que podem ser mapeadas em Diagramas de Classes, de Sequência e de Casos de Uso descrevendo tanto o comportamento dinâmico quanto a estrutura estática do sistema.

Da mesma forma que Diagramas de Objetos, Diagramas de Comunicação permitem uma organização mais livre dos objetos e seus relacionamentos. Essa característica é importante no contexto deste trabalho em situações onde é necessária a representação de ações envolvendo três ou mais mídias. Dessa forma, a representação se torna mais intuitiva e clara para o usuário.

Todos os elementos da UML e seus relacionamentos fazem parte do modelo conceitual da linguagem, também chamado de metamodelo. A partir da extensão da linguagem com elementos de domínio especifico, foi desenvolvido o metamodelo UML-TV (Fig. 2), que define as relações fundamentais entre tais elementos e os elementos do perfil.

Para facilitar o entendimento das relações entre os objetos do metamodelo, duas classes abstratas foram incluídas: *DisplayableMedia* e *Media*. A primeira agrupa objetos que podem ser visualizados, propriedade que pode ser estabelecida a partir do atributo *visible*. A segunda classe abstrata é a generalização de todas as mídias (*Image, Text, Vídeo* e *Audio*).

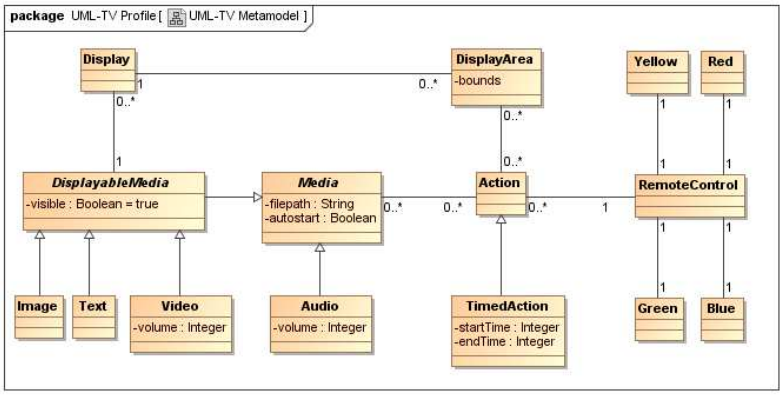

Fig. 2. Metamodelo do perfil UML-TV

A partir do metamodelo, algumas restrições podem ser inferidas como, por exemplo:

1. Uma ação (*Action*) pode se relacionar com uma mídia, com o controle remoto, ou com uma área de visualização.
2. Uma mídia visualizável se relaciona com uma área de visualização através de um elemento *Display*.
3. Uma ação pode se relacionar com apenas um controle remoto, mas o controle remoto pode disparar diversas ações.
4. Considerando que um controle tem apenas uma tecla de cada cor, apenas um conector representando cada uma das teclas coloridas (vermelho, verde, amarelo e azul) pode ser utilizado.

A Fig. 3 mostra os estereótipos definidos e seus respectivos atributos. Nele, pode-se verificar a relação de generalização entre *video*, *text*, *image* e *audio* para *media*, o qual agrupa atributos comuns a todas as mídias (caminho do arquivo – *filepath* – e execução automática – *autostart*).

Todos os elementos passíveis de recepção e/ou envio de mensagens foram classificados como linha de vida (*Lifeline*). Esse é o caso dos elementos *media*, *action*, *remote control* e *display area*.

Já os elementos do perfil que têm significado semântico relacionado à conexão entre os objetos envolvidos, como *GREEN*, *RED*, *YELLOW*, *BLUE*, *timed event* e *display*, foram classificados como extensões do elemento *Connector*. Vale ressaltar que, por motivos de espaço, alguns elementos que são considerados conectores foram

omitidos, como, por exemplo, as demais teclas do controle remoto (números, setas direcionais, etc.).

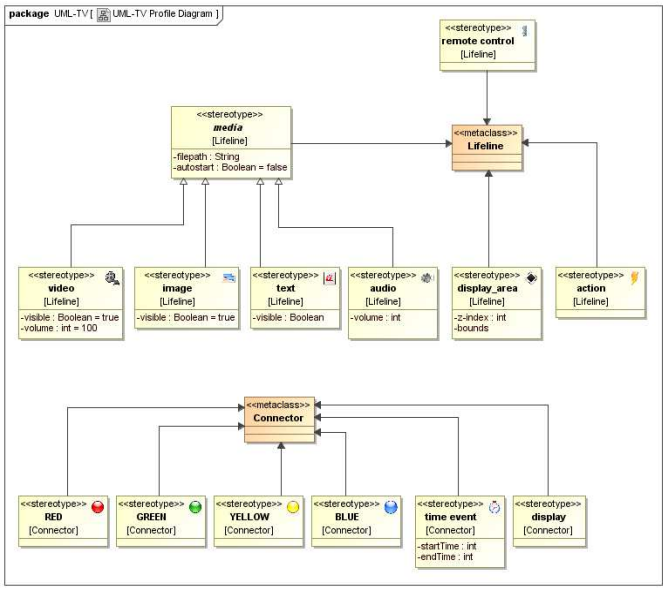

Fig. 3. Especificação do perfil UML-TV

Considerando que o estado dos estereótipos pode mudar durante a execução de uma aplicação, foram criados alguns atributos de forma a possibilitar a correta modelagem destas situações. Por exemplo, o atributo *visible* é vinculado aos estereótipos visualizáveis (*text*, *video* e *image*) com valor padrão *true,* No caso de mídias com som (*video* e *audio*) foi especificado o atributo *volume* com valor padrão *100%.*

O atributo *bounds* coloca uma restrição sobre estereótipo *display area*, especificando o retângulo onde a mídia será apresentada. O retângulo é definido através de quatro valores referentes a um ponto nos eixos x e y, e a altura e largura da região a partir do ponto (x, y).

O estereótipo *timed event* foi enriquecido com a adição de dois atributos, *startTime* e *endTime*, que representam o momento do início e fim da ação, respectivamente.

4. Exemplos de utilização

A partir das propriedades do perfil UML proposto foram modelados alguns exemplos da literatura [13]. Os exemplos ilustrados concentram algumas aplicações em NCL de forma didática e com nível de dificuldade crescente, partindo de uma simples exibição de um vídeo até a construção de um menu de DVD interativo.

4.1. Exemplo 1 – exibindo um vídeo

Nesta seção, é apresentado um exemplo de transformação ilustrando o mapeamento entre o modelo representado em XMI e seu código-fonte da aplicação correspondente em NCL. Neste exemplo, a aplicação exibe um vídeo em determinada área da tela.

A partir do diagrama de comunicação utilizando o perfil UML-TV na Fig. 4, é possível verificar que o vídeo, chamado de *movie,* é iniciado automaticamente e o arquivo de vídeo propriamente dito está localizado em C:\UML-TV\media\video1.mpg. Já a região *movieArea* é demarcada pelo retângulo que tem como ponto superior esquerdo a coordenada (192, 48) e tem largura de 640 pixels por 480 pixels de altura.

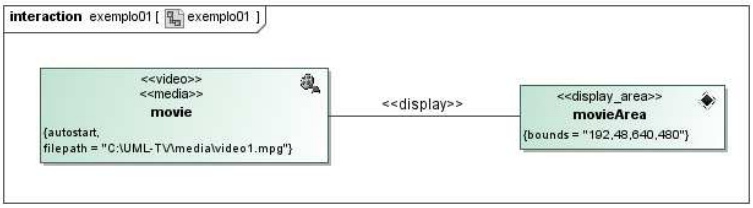

Fig. 4. Exemplo de exibição de um vídeo (*movie*) em uma região (*movieArea*)

A Fig. 5 mostra o código do modelo do exemplo no formato XMI 2.1. Parte do conteúdo do arquivo foi removida, assim como alguns dos identificadores `xmi:id` foram abreviados com o intuito de facilitar a visualização do texto.

A descrição do modelo é iniciada na linha 5, pois esse diagrama foi especificado no mesmo escopo do perfil na ferramenta MagicDraw[2]. As relações entre os elementos são construídas através do cruzamento entre os identificadores.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xmi:XMI xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance" xmlns:UMLTV="http:///schemas/UMLTV/_bYdPUEDrEd6n/0"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:uml=
   "http://www.eclipse.org/uml2/2.0.0/UML" xsi:schemaLocation="http:///schemas/UMLTV/_bYdPUEDrEd6n/0
   UML-TV1.profile.uml#_bkgIRUDrEd6n">
3    <uml:Profile xmi:id="1240163525578_597248_724" name="UML-TV Profile">
4      <packagedElement xmi:type="uml:Collaboration" xmi:id="1240271265702_437536_913" name="exemplos">
5        <ownedBehavior xmi:type="uml:Interaction" xmi:id="912" name="exemplo01">
6          <ownedAttribute xmi:id="935" name="movie" visibility="private" end="704"/>
7          <ownedAttribute xmi:id="678" name="movieArea" visibility="private" end="705"/>
8          <ownedConnector xmi:id="703" name="" visibility="public" kind="delegation">
9            <end xmi:id="704" role="935"/>
10           <end xmi:id="705" role="678"/>
11         </ownedConnector>
12         <lifeline xmi:id="933" name="movie" visibility="public" represents="935"/>
13         <lifeline xmi:id="676" name="movieArea" visibility="public" represents="678"/>
14       </ownedBehavior>
15     </packagedElement>
16   </uml:Profile>
17   <UMLTV:video xmi:id="_bkqfUkDrEd6n" base_Lifeline="933"/>
18   <UMLTV:display_area xmi:id="_bkuwwEDrEd6n" base_Lifeline="676" bounds="192,48,640,480"/>
19   <UMLTV:display xmi:id="_bk36sEDrEd6n" base_Connector="703"/>
20   <UMLTV:media xmi:id="_bk6W80DrEd6n" base_Lifeline="933" filepath="C:\UML-TV\media\video1.mpg"
   autostart="true"/>
21 </xmi:XMI>
```

Fig. 5. Representação do modelo da Fig. 4 em XMI

[2] www.magicdraw.com

Ao se aplicar um *parser* no arquivo, verifica-se que na linha 6, é atribuído o identificador 935 ao vídeo *movie*; na linha 7, o *id* 678 é associado à área *movieArea*. Na linha 8, o conector 703 é criado tem suas pontas definidas como os objetos com *id* 935 e 678, exatamente os objetos *movie* e *movieArea*.

Nas linhas 12 e 13, elementos *lifeline*, que também representam esses dois elementos com *ids* 933 e 676, são criados para que seus atributos sejam associados posteriormente, nas linhas 17, 18 e 20. O resultado da transformação pode ser visto na Fig. 6.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <ncl id="exemplo01"
3  xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
4    <head>
5      <regionBase>
6        <region id="movieArea" left="192" top="48" width="640" height="480" />
7      </regionBase>
8      <descriptorBase>
9        <descriptor id="display1" region="movieArea" />
10     </descriptorBase>
11   </head>
12   <body>
13     <port id="start" component="movie"/>
14     <media type="video/mpeg" id="movie" src="C:\UML-TV\media\video1.mpg" descriptor="display1"/>
15   </body>
16 </ncl>
```

Fig. 6. Código gerado a partir da representação do modelo em XMI

4.2. Exemplo 2 – exibição de video com interatividade

No modelo da Fig. 7, dois vídeos chamados *video1* e *video2* são executados ao mesmo tempo e iniciam automaticamente quando a aplicação começa a ser executada. Ambos são mostrados na mesma área da tela (*rgVideo*). No entanto, suas propriedades *visible* e *volume* são diferentes. A mídia *video1* possui os valores *true* e *100,* enquanto que a mídia *video2* tem os valores *false* e *0* (zero). Isso significa que, apesar de estarem sendo executados ao mesmo tempo, apenas o *video1* pode ser visto e ouvido.

Além dos dois vídeos acima descritos, a imagem *redButton* também é mostrada em uma área determinada chamada de *rgRedButton* assim que a aplicação é iniciada. A imagem *greenButton*, associada à área *rgGreenButton* também é especificada, mas não pode ser visualizada inicialmente. Vale ressaltar que algumas informações como o tipo e o caminho no sistema de arquivos, inerentes às mídias, foram omitidas por não serem interessantes para o estudo de caso em questão.

Ao se iniciar o documento hipermídia, o usuário visualizará duas mídias: o vídeo *video1* e a imagem *redButton*. Nesse momento, caso o usuário pressione o botão vermelho do controle remoto, a ação chamada *red* será ativada e os eventos que partem desse objeto serão executados. Desta forma, a exibição da imagem *redButton* será finalizada, a exibição da imagem *greenButton* será iniciada na região *rgGreenButton*, o vídeo *video1* terá seus atributos *visible* e *volume* alterados para *false* e *0* (zero), respectivamente, assim como os mesmos atributos do vídeo *video2* serão alterados para *true* e *100*. Similarmente, ao se pressionar o botão verde de controle

remoto, os eventos associados à ação *green* serão ativados e realizarão o processo inverso ao da ação *red*.

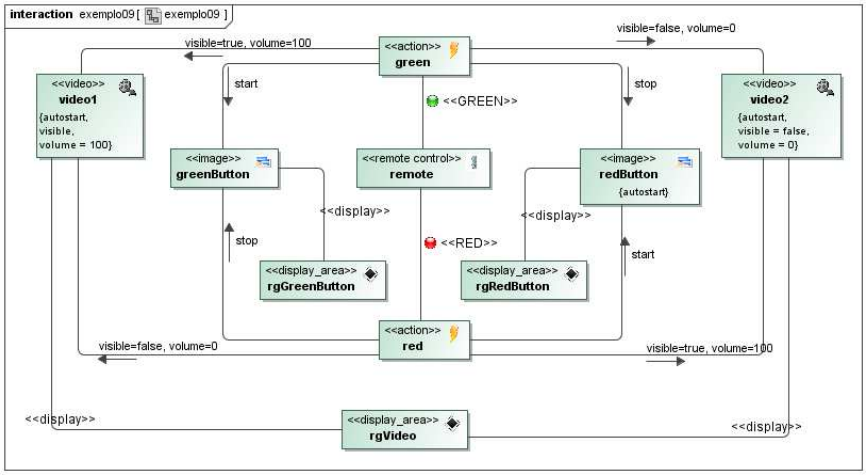

Fig. 7. Modelo de uma aplicação onde dois vídeos são executados simultaneamente

Essa aplicação pode ser utilizada em um contexto maior para, por exemplo, permitir que o usuário, a partir do controle remoto, interaja escolhendo qual ângulo da câmera deseja visualizar durante determinado programa. A Fig. 8 mostra o estado inicial da aplicação.

Fig. 8. Aplicação modelada sendo executada

Toda a modelagem foi realizada com a ferramenta MagicDraw a qual permite a exportação dos modelos no formato XMI (*XML Metadata Interchange*) [10]. A implementação da transformação da aplicação modelada para o código-fonte da aplicação, neste caso um documento NCL, foi realizada através de transformações XSLT (*XSL Transformations*) [5] onde o documento XMI pôde ser mapeado para um documento NCL.

Como pode ser notado, a transformação foi aplicada do modelo independente de plataforma, PIM (*Platform Independent Model*), diretamente para o código-fonte. No entanto, um modelo específico de plataforma (PSM – *Platform Specific Model*) para a linguagem NCL já está sendo desenvolvido com o intuito de facilitar a geração de aplicações mais complexas para essa linguagem.

5. Conclusão e trabalhos futuros

O início da operação da TV digital no Brasil e a adoção do padrão brasileiro por outros países da América do Sul se mostra como um momento ideal para o surgimento de novas idéias que venham a contribuir com a qualidade das aplicações desenvolvidas. O perfil UML-TV, aliado a uma boa ferramenta CASE de modelagem UML, se constitui em uma nova e eficiente alternativa para a produção de conteúdo para televisão digital.

Várias das vantagens da utilização do perfil UML-TV são inerentes aos conceitos de arquitetura orientada a modelos. Entre eles a possibilidade de manter a documentação das aplicações atualizada com o código-fonte já que as alterações são realizadas diretamente no modelo e então transformadas, a possibilidade maior de entendimento da modelagem por envolvidos que não têm conhecimento técnico sobre o assunto e o aproveitamento do conhecimento sobre a notação UML, que faz com que o tempo de aprendizado seja menor.

O perfil UML-TV possui relativamente poucas adições de estereótipos e atributos, a partir dos quais é possível cobrir diferentes aplicações utilizando diversos tipos de sincronismo entre mídias. Esta propriedade possibilita a modelagem de uma grande variedade de aplicações hipermídia interativas e não interativas.

Como trabalhos futuros, uma melhor formalização das restrições do perfil será realizada, assim como a especificação de modelos específicos de domínio (PSM) e as respectivas transformações M2M (*model to model*). Também serão realizados testes com aplicações mais complexas e de uso em ambientes reais como forma de validar o trabalho mais efetivamente. Com isso, outros refinamentos no metamodelo e no perfil também podem ser realizados, com a adição de novos estereótipos e atributos.

Referências

1. Aagedal, J. O., Ecklund, E. F.: Modelling QoS: Towards a UML Profile. Lecture Notes in Computer Science (2002)
2. Azevedo, R.: NCL Eclipse. http://laws.deinf.ufma.br/~ncleclipse (2008)
3. Barnaghi, P. M., Kareem, S. A.: UML Extensions for Hypermedia Navigation and Presentation Modeling. (2008)
4. Bertino, E., Ferrari, E., Stolf, M.: MPGS: an Interactive Tool for the Specification and Generation of Multimedia Presentations. IEEE Transactions on Knowledge and Data Engineering, 12(1):102–125 (2000)
5. Clark, J.: XSL Transformations (XSLT) version 1.0. W3C Recommendation, 16(11) (1999)

6. Fontoura, M., Pree, W., Rumpe, B.: The UML Profile for Framework Architectures. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (2000)
7. Fórum do Sistema Brasileiro de TV Digital Terrestre. O que é o ISDB-TB. http://www.forumsbtvd.org.br/materias.asp?id=20 (2009)
8. Guimarães, R. L., Costa, R., Soares, L. F. G.: Composer: Authoring Tool for iTV Programs. Proceedings of the 6th European Conference on Changing Television Environments, pages 61–71. Springer-Verlag Berlin, Heidelberg (2008)
9. Júnior, J. U., Camargo, V. V., Chavez, C. V. F.: UML-AOF: a Profile for Modeling Aspect-Oriented Frameworks. Proceedings of the 13th Workshop on Aspect-Oriented Modeling. Charlottesville, Virginia, USA: ACM (2009)
10. Object Management Group. XML Metadata Interchange. http://www.omg.org/technology/documents/formal/xmi.htm (2007)
11. Sauer, S., Engels, G.: UML-based Behavior Specification of Interactive Multimedia Applications. Proceedings IEEE Symposia on Human-Centric Computing Languages and Environments, pages 248–255 (2001)
12. Soares, L. F. G., Rodrigues, R. F., Moreno, M. F.: Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System. Journal of the Brazilian Computer Society, 12(4):37–46 (2007)
13. Soares Neto, C. S., Soares, L. F. G., Rodrigues, R. F., Junqueira, S. D.: Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer. Relatório Técnico, PUC-Rio (2007)
14. Soley, R.: Model Driven Architecture. White paper, November 27 (2001)
15. de Souza Filho, G. L., Leite, L. E. C., Batista, C.: Ginga-J: the Procedural Middleware for the Brazilian Digital TV System. Journal of the Brazilian Computer Society, 13(4):47–56 (2007)
16. Ziadi, T., Helouet, L., Jezequel, J. M.: Towards a UML Profile for Software Product Lines. Lecture Notes in Computer Science (2004)