6
Referências Bibliográficas

[Adobe, 2007a] Adobe Systems, **Dreamweaver 8**. Disponível em http://www.macromedia.com/software/dreamweaver/. Acesso em 22 de fevereiro de 2007.

[Adobe, 2007b] Adobe Systems, **Macromedia Flash MX Professional 7** – Disponível em http://www.adobe.com/products/flash/flashpro/. Acesso em 27 de fevereiro de 2007.

[Alder, 2003] Alder, G., **Design and Implementation of the JGraph Swing Component**, Technical Report, Fevereiro de 2003. Disponível em http://ontwerpen1.khlim.be/projects/netsim/jgraph-paper.pdf. Acesso em 16 de março de 2007.

[Alticast, 2007a] Alticast Inc, **AltiComposer 2.0** - USA. Disponível em http://www.alticast.com. Acesso em 22 de fevereiro de 2007.

[Alticast, 2007b] Alticast Inc, **Script Guide - AltiComposer 2.0**, USA. Disponível em http://www.alticast.com. Acesso em 27 de fevereiro de 2007.

[Alticast, 2007c] Alticast Inc, **User Guide – AltiComposer 2.0**, USA. Disponível em http://www.alticast.com. Acesso em 27 de fevereiro de 2007.

[Berners-Lee et al., 1994] Berners-Lee, T. J.; Cailliau, R.; Luotonen, A.; Nielsen, H. F.; Secret, A., **The World-Wide Web**, Communications of the ACM, v. 37, n. 8, Agosto de 1994, p. 76-82. Disponível em http://portal.acm.org/citation.cfm?id=179671. Acesso em 3 de abril de 2007.

[Buchanan & Zelweger, 1992] Buchanan, M. C.; Zelweger, P. T., **Specifying Temporal Behavior in Hypermedia Documents**, European Conference on Hypertext - ECHT'92, Milão, Itália, Dezembro de 1992. Disponível em http://citeseer.ist.psu.edu/cache/papers/cs/4389/http:zSzzSzwww.parc.xerox.comzSzistlzSzmemberszSzpollezzSzpaperszSzfirefly-echt92.pdf/buchanan92specifying.pdf. Acesso em 18 de março de 2007.

[Bulterman & Hardman, 2005] Bulterman, D.; Hardman, L., **Structured Multimedia Authoring**, ACM Transactions on Multimedia Computing Communications, and Applications (TOMCCAP), v. 1, i. 1, p. 89-109, Fevereiro de 2005. Disponível em http://portal.acm.org/citation.cfm?id=1047943&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618. Acesso em 18 de março de 2007.

[Bulterman & Rutledge, 2004] Bulterman, D.; Rutledge, L., Livro **SMIL 2.0: Interactive Multimedia for Web and Mobile Devices**. Springer, Abril de 2004.

[Bulterman et al., 1998] Bulterman, D. C. A.; Hardman, L.; Jansen, J.; Mullender, K. S.; and Rutledge, L. **GRiNS: A GRaphical INterface for creating and playing SMIL documents.** In WWW7 Conference, Computer Networks and ISDN Systems, v. 30, i. 1-7, p. 519-529, Brisbane, Australia, Abril de 1998. Disponível em http://epubs.cclrc.ac.uk/bitstream/428/GRiNS_final.pdf. Acesso em 20 de abril de 2007.

[CableLabs, 2006] CableLabs, **OpenCable Application Platform Specification (OCAP) 1.1**. Issued I01, Dezembro de 2006. Disponível em http://www.opencable.com/downloads/specs/OC-SP-OCAP1.1-I01-061229.pdf. Acesso em 23 de fevereiro de 2007.

[Canonical, 2007] Canonical Ltd, **Ubuntu 6.10 (Linux distribution)**. Disponível em http://www.ubuntu.com/. Acesso em 23 de maio de 2007.

[Cardinal, 2007] Cardinal Information Systems Ltd, **Cardinal Studio 4.0** – Finland. Disponível em http://www.cardinal.fi. Acesso em 22 de fevereiro de 2007.

[Coelho & Soares, 2004] Coelho, R. M.; Soares, L. F. G., **Integração de Ferramentas Gráficas e Declarativas na Autoria de Arquiteturas Modeladas através de Grafos Compostos** - Dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio - 2004. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2004_08_coelho.pdf. Acesso em 18 de março de 2007.

[Costa & Soares, 1996] Costa, F. R.; Soares, L. F. G., **Um Editor Gráfico para Definição e Exibição do Sincronismo de Documentos Multimídia/Hipermídia**, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Agosto de 1996. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/1996_08_COSTA.pdf. Acesso em 3 de abril de 2007.

[Costa & Soares, 2005] Costa, R. M. R.; Soares, L. F. G., **Modelos Temporais para Especificação de Sincronismo em Documentos Hipermídia**, monografia apresentada como requisito de avaliação da disciplina Estudo Orientado, 2º semestre de 2005.

[Costa et al., 2006] Costa, R. M. R.; Moreno, M. F.; Rodrigues, R. F.; Soares, L. F. G., **Live Editing of Hypermedia Documents**, Proceedings of the 2006 ACM symposium on Document engineering - DocEng'06, Amsterdam, The Netherlands, 2006. Disponível em http://delivery.acm.org/10.1145/1170000/1166202/p165-costa.pdf?key1=1166202&key2=1513054711&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618. Acesso em 21 de março de 2007.

[ECMA, 1999] ECMA Standardizing Information and Communication Systems. **ECMAScript Language Specification**, Standard ECMA 262, 3rd Edition, Dezembro de 1999. Disponível em http://www.ecma-international.org/publications/standards/Ecma-262.htm. Acesso em 22 de fevereiro de 2007.

[ETSI, 2003] ETSI, ES 201 812: **Multimedia Home Platform (MHP) Specification 1.0.3**. ETSI Standard, Dezembro de 2003. Disponível em http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=18800. Acesso em 22 de fevereiro de 2007.

[FLEXTV, 2004] FLEXTV, **TV Digital: Análise das Alternativas Tecnológicas**. Relatório em atendimento ao Requisito 4.1.3 – RFP04/2004 – MCT/FINEP, produto 1A, Dezembro de 2004.

[Fraunhofer, 2007a] Fraunhofer Institute for Media Communication IMK, **JAME Author 1.0**, Schloss Birlinghoven – Germany. Disponível em http://www.jame.tv. Acesso em 22 de fevereiro de 2007.

[Fraunhofer, 2007b] Fraunhofer Institute for Media Communication IMK, **Quick Start - JAME Author 1.0**, Schloss Birlinghoven - Germany. Disponível em http://www.jame.tv. Acesso em 27 de fevereiro de 2007.

[Fraunhofer, 2007c] Fraunhofer Institute for Media Communication IMK, **User Manual - JAME Author 1.0**, Schloss Birlinghoven – Germany. Disponível em http://www.jame.tv. Acesso em 27 de fevereiro de 2007.

[FSF, 2007] Free Software Foundation, **GNU Lesser General Public License**. Disponível em http://www.gnu.org/copyleft/lgpl.html. Acesso em 23 de março de 2007.

[Gamma et al., 1997] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley Publishing Company, 11a. edição, Maio de 1997.

[Hardman et al., 1994] Hardman, L.; Bulterman, D. C. A.; Van Rossum, G., **The Amsterdam hypermedia model: adding time and context to the Dexter model**, Communications of the ACM, v. 37, i. 2, 1994, p. 50-62. Disponível em http://citeseer.ist.psu.edu/cache/papers/cs/17548/ftp:zSzzSzftp.cwi.nlzSzpubzSzmmpaperszSzcacm.pdf/the-amsterdam-hypermedia-model.pdf. Acesso em 23 de abril de 2007.

[ICE-CREAM, 2007] ICE-CREAM Consortium, **Interactive Consumption of Entertainment in Consumer Responsive, Engaging & Active Media**. Disponível em http://www.hitech-projects.com/euprojects/icecream. Acesso em 23 de fevereiro de 2007.

[ISO, 1998] ISO/IEC TR 13818-6. **Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC,** ISO Standard, 1998. Disponível em http://www.iec.ch/cgi-bin/getcorr.pl/isoiec13818-6-amd1-cor1%7Bed1.0%7Den.pdf?file=isoiec13818-6-amd1-cor1%7Bed1.0%7Den.pdf. Acesso 18 de março de 2007.

[ISO, 2000] ISO/IEC 13818-1. **Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems**. ISO Standard, 2000. Disponível em http://webstore.iec.ch/preview/info_isoiec13818-1%7Bed2.0%7Den.pdf. Acesso em 29 de março de 2007.

[ISO, 2001] ISO/IEC 14496-1. **Information technology - Coding of audio-visual objects - Part 1: Systems,** ISO Standard, 2001. Disponível em http://webstore.ansi.org/ansidocstore/product.asp?sku=INCITS%2FISO%2FIEC+14496-1-2001%2FAM1-2001. Acesso em 18 de março de 2007.

[ISO, 2006] ISO/IEC 19757-3. **Information technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron**, ISO Standard, 2006. Disponível em http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip. Acesso em 17 de maio de 2007.

[Kamada & Kawai, 1989] Kamada, T.; Kawai, S., **An algorithm for drawing general undirected graphs**, Information Processing Letters, v. 31, i. 1, p. 7-15, Abril, 1989.

[Lamdon et al., 2003] Lamdon, J. L.; Cesar, P.; Herrero, C.; Vuorimaa, P., **Usages of a SMIL player in digital television**, Proceedings of the VII International Conference on Internet and Multimedia Systems and Applications – IASTED, USA, Agosto de 2003. Disponível em http://lib.tkk.fi/Diss/2005/isbn951227888X/article5.pdf. Acesso em 18 de março de 2007.

[Microsoft, 2007a] Microsoft Corporation, **Microsoft FrontPage**. Disponível em http://www.microsoft.com/frontpage/. Acesso em 22 de fevereiro de 2007.

[Microsoft, 2007b] Microsoft Corporation, **Microsoft Windows XP Professional**. Disponível em http://www.microsoft.com/windowsxp/pro/default.mspx. Acesso em 23 de maio de 2007.

[Microsoft, 2007c] Microsoft Corporation, **Windows Internet Explorer**. Disponível em http://www.microsoft.com/windows/products/winfamily/ie/default.mspx. Acesso em 25de abril de 2007.

[Moreno et al., 2005] Moreno, M. F.; Costa, R. M. R.; Rodrigues, R. F.; Soares, L. F. G., **Edição de Documentos Hipermídia em Tempo de Exibição**, X Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, Poços de Caldas, Brasil, Dezembro de 2005. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/2005_12_moreno_webmedia.pdf. Acesso em 18 de março de 2007.

[Moura & Soares, 2001] Moura, M. S. A.; Soares, L. F. G., **Relações Espaciais em Documentos Hipermídia**, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Agosto de 2001. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2001_08_moura.pdf. Acesso em 3 de abril de 2007.

[Muchaluat-Saade et al., 2003] Muchaluat-Saade, D. C.; Silva, H. V. O; Soares, L. F. G., **Linguagem NCL versão 2.0 para Autoria Declarativa de Documentos Hipermídia**, IX Simpósio Brasileiro de Sistemas Multimídia e WEB - WebMídia 2003, Salvador, Brasil, Novembro de 2003. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/2003_11_muchaluat_webmidia.pdf. Acesso em 18 de março de 2007.

[Pemberton et al., 2002] Pemberton, S.; Austin, D.; Axelsson, J.; et al. **XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)**, 2002. Disponível em http://www.w3.org/TR/xhtml1/. Acesso em 22 de fevereiro de 2007.

[Peng & Vuorimaa, 2001] Peng, C.; Vuorimaa, P., **Digital Television Application Manager**, Proceedings of the IEEE International Conference on Multimedia and Expo, 2001, Japão, Agosto de 2001. Disponível em http://lib.tkk.fi/Diss/2002/isbn9512261723/article8.pdf. Acesso em 18 de março de 2007.

[Peng et al., 2001] Peng, C.; Cesar, P.; Vuorimaa, P., **Integration of Applications into Digital Television Environment**, Proceedings of the VII International Conference on Distributed Multimedia Systems, Taiwan, Setembro de 2001. Disponível em http://lib.tkk.fi/Diss/2002/isbn9512261723/article7.pdf. Acesso em 18 de maio de 2007.

[Pestov, 2007] Pestov, S., **JEdit - Programmer's Text Editor**. Disponível em http://www.jedit.org/. Acesso em 15 de março de 2007

[Pihkala et al., 2002] Pihkala, K.; Cesar, P.; Vuorimaa, P., **A Cross-Platform SMIL Player**, Procedings of the International Conference on Communication, Internet and Information - IASTED, USA, Novembro de 2002. Disponível em http://lib.tkk.fi/Diss/2003/isbn9512268043/article7.pdf. Acesso em 18 de março de 2007.

[Rodrigues & Soares, 2003] Rodrigues, R. F.; Soares, L. F. G., **Formatação e controle de apresentações hipermídia com mecanismos de adaptação temporal.** Rio de Janeiro, 2003. 156p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro. Disponível em ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2003_03_rodrigues.pdf. Acesso em 03 de abril de 2007.

[Silva et al., 2004] Silva, H. V.; Rodrigues, R. F.; Soares, L. F. G., **Frameworks para Processamento de Linguagens XML Modulares**, Relatório Técnico, Laboratório TeleMídia, PUC-Rio, Brasil, 2004.

[Soares & Rodrigues, 2006] Soares, L. F. G.; Rodrigues, R. F., **Nested Context Language 3.0 - Part 8 – NCL Digital TV Profiles**. Monografias em Ciência da Computação, n? 35/06, Departamento de Informática, PUC-Rio, Brasil, Outubro de 2006. Disponível em http://www.ncl.org.br/documentos/NCL3.0-DTV.pdf. Acesso em 24 de abril de 2007.

[Soares et al., 2003] Soares, L. F. G.; Rodrigues, R. F.; Muchaluat-Saade, D. C., **Modelo de Contextos Aninhados – versão 3.0**, Relatório Técnico, Laboratório TeleMídia, Departamento de Informática, PUC-Rio, 2003.

[Soares et al., 2004] Soares, L. F. G.; Colcher, S., Rodrigues; R. F., **Relatório TV Digital: Identificação dos Cenários Tecnológicos de Interesse**, Laboratório TeleMídia, PUC-Rio, 2004.

[Soares et al., 2006] Soares, L. F. G.; Rodrigues, R. F.; Costa, R. M. R., **Part 6 - NCL (Nested Context Language) Main Profile**. Monografia nº 15/06, Departamento de Informática, PUC-Rio, Abril de 2006. Disponível em http://www.ncl.org.br/documentos/ncl23-mainprofile.pdf. Acesso em 12 de março de 2007.

[Sun, 1994] Sun Microsystems, **Java Technology**. Disponível em http://www.java.sun.com. Acesso em 22 de fevereiro de 2007.

[Sun, 1997] Sun Microsystems, **Java Native Interface (JNI)**. Disponível em http://java.sun.com/j2se/1.5.0/docs/guide/jni/. Acesso em 23 de março de 2007.

[Sun, 2000] Sun Microsystems, **Java TV API**. Disponível em http://java.sun.com/products/javatv/. Acesso em 24 de fevereiro de 2007.

[Sun, 2003] Sun Microsystems, **JavaHelp System**. Disponível em http://java.sun.com/products/javahelp/index.jsp. Acesso em 15 de abril de 2007.

[Vignoni, 2007] Vignoni, D., **Nuvola 1.0 for KDE 3.x**. Disponível em http://icon-king.com/?p=15. Acesso em 14 de março de 2007.

[W3C, 1999a] W3C - World-Wide Web Consortium, **HTML 4.01 Specification**. W3C Recommendation, Dezembro de 1999. Disponível em http://www.w3.org/TR/html401/. Acesso em 22 de fevereiro de 2007.

[W3C, 1999b] W3C - World-Wide Web Consortium, **Hypertext Transfer Protocol -- HTTP/1.1**. RFC2616, Junho de 1999. Disponível em http://www.w3.org/Protocols/rfc2616/rfc2616.html. Acesso em 24 de abril de 2007.

[W3C, 2001] W3C - World-Wide Web Consortium, **Synchronized Multimedia Integration Language (SMIL 2.0) Specification**, W3C Recommendation, Agosto de 2001. Disponível em http://www.w3.org/TR/smil20/. Acesso em 23 de fevereiro de 2007.

[W3C, 2004a] W3C - World-Wide Web Consortium, **Extensible Markup Language (XML) 1.1**, Fevereiro de 2004. Disponível em http://www.w3.org/TR/2004/REC-xml11-20040204/. Acesso em 22 de fevereiro de 2007.

[W3C, 2004b] W3C - World-Wide Web Consortium, **Cascading Style Sheets (CSS)**, Fevereiro de 2004. Disponível em http://www.w3.org/Style/CSS/. Acesso em 22 de fevereiro de 2007.

[W3C, 2004c] W3C - World-Wide Web Consortium, **Document Object Model (DOM) Level 3**, W3C Recommendation, Abril de 2004. Disponível em http://www.w3.org/TR/DOM-Level-3-Core/. Acesso em 22 de fevereiro de 2007.

[WAM, 2007] WAM (Web Adaptation Multimedia), **LimSee2 - The cross-platform SMIL2.0 authoring tool**, INRIA - Grenoble, France. Disponível em http://wam.inrialpes.fr/software/limsee2/. Acesso em 27 de fevereiro de 2007.

[Williams, 2002] Williams, M., **ActionScript Coding Standards**, Macromedia White Paper, Março de 2002. Disponível em http://www.adobe.com/devnet/flash/whitepapers/actionscript_standards.pdf. Acesso em 24 de abril de 2007.

A
Linguagens, Sincronismo e Interatividade

Este capítulo está organizado da seguinte forma. A Seção A.1 apresenta como as linguagens Java e HTML (XHTML) são adotadas no desenvolvimento de aplicações voltadas para TV digital. A Seção A.2 apresenta algumas linguagens que representam modelos hipermídia, as linguagens declarativas, e como elas podem ser usadas pelos padrões de TV digital interativa.

A.1.
Linguagens nos padrões de TV digital

Os padrões atuais de TV digital (Soares et al., 2004) oferecem linguagens para a especificação da interatividade e do sincronismo nas aplicações. Em comum, as aplicações especificadas em Java, e em XHTML + linguagens de scripts, como ECMAScript (ECMA, 1999), apresentam restrições à autoria, pois requerem conhecimentos de programação para serem utilizadas no desenvolvimento de aplicações. As Subseções A.1.1 e A.1.2 apresentam como as linguagens Java e XHTML, respectivamente, são adotadas pelos principais padrões de TV digital, apresentando suas características.

A.1.1.
Linguagens baseadas em Java

As linguagens DVB-J, ACAP-J e ARIB-B23, relativas, respectivamente, aos padrões DVB (*Digital Video Broadcast*), ATSC (*North-America/US Advanced Television Systems Committee*) e ISDB (*Integrated Services Digital Broadcasting*) (Soares et al., 2004), têm como principal diferença os conjuntos de classes que cada padrão implementa. Em todas essas linguagens, que são linguagens procedurais baseadas na tecnologia Java, sem exceção, o programador deve modelar o conjunto de objetos da aplicação e a relação entre esses objetos.

Na especificação dos objetos através de uma linguagem orientada a objetos, é função do programador definir a estrutura de dados interna de cada objeto e as

interfaces que serão utilizadas na comunicação com os outros objetos da aplicação (encapsulamento). Nesse paradigma de linguagem, objetos podem ser instanciados a partir de um conjunto hierárquico de classes definidas pelo *middleware*, herdando a estrutura de dados dessas classes e suas respectivas interfaces (herança). Além disso, durante a execução, um objeto pode ser instanciado a partir de uma das diversas classes dentro da hierarquia na qual ele foi definido, adquirindo um comportamento distinto de acordo com a instância escolhida (polimorfismo).

Nas linguagens DVB-J, ACAP-J e ARIB-B23, as relações entre os objetos são especificadas através de métodos, onde o programador deve especificar uma sucessão de passos a serem seguidos utilizando estruturas de controle, variáveis e chamadas a outros métodos. As linguagens desses padrões são, portanto, baseadas em uma linguagem de programação de propósito geral, onde o programador dispõe de muitos recursos para a especificação da aplicação. Em contrapartida, existe uma razoável complexidade na especificação dessas aplicações.

Por assim serem, as linguagens DVB-J, ACAP-J e ARIB-B23, não apresentam mecanismos diretos para a especificação de sincronismo e interatividade na construção de aplicações hipermídia. Essas especificações devem ser realizadas pelo programador de acordo com os métodos existentes nos conjuntos de classes (pacotes) oferecidas pelo *middleware*.

Entre os conjuntos de classes que podem ser encontrados nos diversos *middlewares* dos padrões DVB, ATSC e ISDB, podem ser destacadas as classes para o desenvolvimento de interfaces gráficas AWT (*Abstract Window Toolkit*) e JMF (*Java Media Framework*), além das classes para transmissão de dados DAVIC (*Digital Audio Video Interactive Consortium*) e para interconexão de dispositivos HAVi (*Home Audio/Video Interoperability*). Maiores detalhes sobre a especificação desses pacotes podem ser encontrados no relatório "Análise das Alternativas Tecnológicas" (FLEXTV, 2004).

Os pacotes da implementação JavaTV (Sun, 2000) também são usualmente encontrados nos *middlewares* dos padrões de TV digital. As classes definidas no pacote *javax.tv.xlet* merecem destaque por implementarem o ciclo de vida das aplicações. As aplicações criadas a partir das classes definidas nesse pacote são genericamente denominadas *Xlets*. Os *Xlets* são controlados por um gerenciador de aplicações (*application manager*) que faz parte do *middleware* e reside no *set-*

top box (Peng & Vuorimaa, 2001). As principais funções do gerenciador de aplicações são: sinalizar as mudanças de estados dos *Xlets* e controlar os recursos do *middleware* a partir da programação do *Xlet*.

Os *Xlets* podem estar residentes no *set-top box* ou serem recebidos por carrossel (transporte cíclico) de objetos ou de dados (Soares et al., 2004). Todo o controle dos *Xlets* é realizado pelo gerenciador de aplicações, incluindo a iniciação, que pode ocorrer de duas formas: o *Xlet* pode estar temporalmente sincronizado com o fluxo de transporte MPEG-2 (ISO, 2000), correspondente ao fluxo de áudio e vídeo principal, ou pode ser iniciado pela ação do usuário. Em ambas as formas, é importante que os *Xlets* possuam baixa latência de iniciação (Peng et al., 2001), a fim de preservar a sincronização da aplicação com o conteúdo audiovisual principal.

A Figura 33 apresenta os quatro estados de um *Xlet*: carregado (*loaded*), pausado (*paused*), iniciado (*started*) e destruído (*destroyed*) (Soares et al., 2004). O gerenciador de aplicações é capaz de controlar múltiplas aplicações simultâneas, no entanto, somente uma aplicação é visível em um dado instante de tempo. No contexto dos estados, somente um *Xlet*, em um dado instante de tempo, pode se encontrar no estado iniciado.

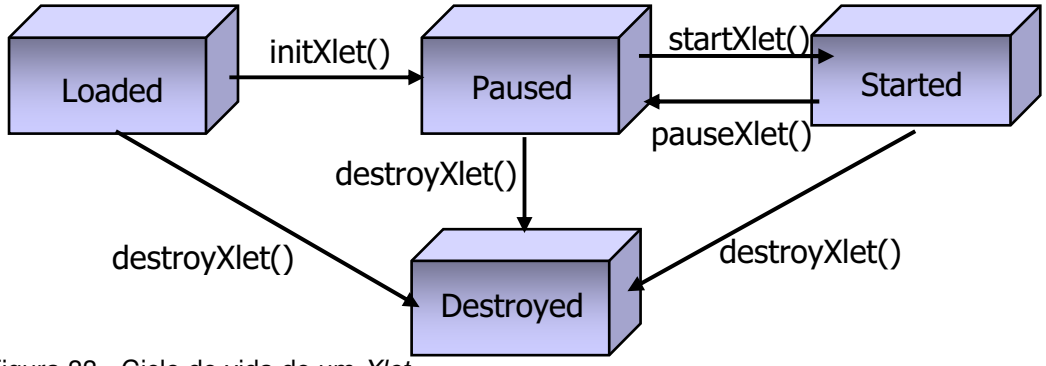

Figura 33 - Ciclo de vida de um *Xlet*

Quando o gerenciador de aplicações recebe um novo *Xlet*, o construtor desse *Xlet* é executado. Durante a execução do construtor, o *Xlet* encontra-se no estado carregado. Posteriormente, quando uma das condições de iniciação do *Xlet* ocorre, o método *initXlet()* da aplicação é executado, tendo como parâmetro o contexto[9]

[9] As informações do contexto incluem, principalmente, os recursos alocados por um *Xlet*, como arquivos ou acesso a dispositivos de entrada e saída. Esses recursos somente são liberados quando o *Xlet* é destruído.

do *Xlet*. Nesse momento, o *Xlet* passa para o estado pausado. A partir desse ponto, quando o método *startXlet()* é executado, o *Xlet* passa para o estado iniciado. A qualquer momento, durante o estado iniciado de um *Xlet*, o gerenciador de aplicações pode solicitar a execução do método *pauseXlet()* e retornar a aplicação para o estado pausado. Quando um *Xlet* termina sua execução, o seu método *destroyXlet()* é executado e os recursos alocados são liberados. As condições de término de um *Xlet* são idênticas às condições de iniciação.

A.1.1.1. Sincronismo e Interatividade nas linguagens baseadas em Java

Conforme mencionado no início deste capítulo, para definir o sincronismo e interatividade em aplicações através da linguagem Java, é necessário que o autor tenha conhecimentos de programação. Nessa linguagem, os eventos de sincronismo e interatividade, bem como os demais eventos que podem ocorrer em uma aplicação multimídia, são, em sua maioria, implementados através de observadores (*listeners*). Esses observadores são instanciados nos *Xlets* que constituem as aplicações.

Os observadores nas aplicações Java para TV digital podem ser utilizados desde o início da apresentação. Para receber o conteúdo dos objetos, preservando o controle sobre a aplicação, os *Xlets* podem utilizar *threads* Java para realizar o carregamento do conteúdo dos objetos de mídia de forma assíncrona.

Como exemplo de acesso assíncrono aos objetos de mídia, a linguagem DVB-J, através da classe *org.dvb.dsmcc.DSMCCObject*, implementa o acesso aos objetos transmitidos por carrossel. A classe *DSMCCObject* deve receber no seu construtor o objeto Java que instancia o carrossel e o caminho (*path*) do objeto de mídia solicitado pela aplicação. O acesso ao objeto de mídia pode ser feito de forma assíncrona utilizando o método *asynchronousLoad()*, do objeto da classe *DSMCCObject*. Esse método recebe como parâmetro um observador, cuja classe é definida a partir da especificação da classe *java.util.EventListener*. Essa classe deve ser projetada para avisar a aplicação (*Xlet*), quando o objeto de mídia estiver carregado.

Para implementar os pontos de sincronização pertencentes a um fluxo (*stream events*), as linguagens baseadas em Java também utilizam observadores.

Como exemplo, na linguagem DVB-J, o controle dos *stream events* é realizado pela classe *org.dvb.dsmcc.DSMCCStreamEvent*. Para instanciar um objeto dessa classe, deve ser passado como parâmetro ao seu construtor, um objeto da classe *DSMCCObject*. Nesse caso, a aplicação irá monitorar os pontos de sincronização que forem recebidos e que estejam relacionados ao objeto passado como parâmetro ao construtor da classe.

Para cada tipo de *stream event* definido pela aplicação, deve ser definido um observador correspondente. Os observadores, definidos a partir da especificação da classe *java.util.EventListener*, devem ser associados aos tipos de *stream events* através do método *subscribe()* do objeto da classe *DSMCCStreamEvent*. Caso um tipo de *stream event* seja considerado irrelevante pela aplicação, o observador a ele associado deve ser desativado através da chamada ao método *unsubscribe()*.

Além dos eventos de sincronização do DSM-CC (*Digital Storage Media Command and Control*) (ISO, 1998), as aplicações para TV digital, especificadas através das linguagens DVB-J, ACAP-J e ARIB-B23 podem capturar eventos de interatividade originados pelo usuário. Para definir esses eventos, podem ser utilizadas as classes definidas no pacote HAVi, como por exemplo, a classe *org.havi.ui.event.HRcEvent* que especifica os eventos das teclas do controle remoto.

Para exemplificar o controle de eventos relacionados à interatividade, considere um aplicativo (*Xlet*) na linguagem DVB-J, que instancia um repositório relativo aos tipos de teclas cujo pressionamento deve ser monitorado. Esse repositório é instanciado através da classe *org.dvb.event.UserEventRepository*, e pode, por exemplo, conter os eventos relativos às teclas coloridas, através do método *addAllColorKeys()*, ou os eventos relativos às teclas numéricas, através do método *addAllNumericKeys()*, bem como outros eventos, relativos às demais teclas encontradas em controles de televisores.

Nesse *Xlet*, a partir da definição do repositório, deve ser definido um gerenciador de eventos (*event manager*) para controlar, através de um observador, a ocorrência dos eventos relativos às teclas definidas no repositório. O gerenciador de eventos é instanciado através do método estático *getInstance()*, definido na classe *org.dvb.EventManager*. O observador, que implementa a classe *org.dvb.UserEventListener*, deve ser passado como parâmetro ao objeto da classe *EventManager*, através do método *addUserEventListener()*. Os eventos retornados

ao observador são definidos na classe *org.dvb.event.UserEvent*. A Figura 34 apresenta, como exemplo, um trecho de uma classe que especifica o controle de eventos no DVB-J.

```
class App implements UserEventListener{

  public App () {
    EventManager em ;
    UserEventRepository repository ;
    em = EventManager.getInstance () ;
    repository = new UserEventRepository ("R1") ;
    repository.addKey (UserEvent.VK_ENTER) ;
    em.addUserListener ((UserEventListener) this, repository) ;
    em.addResourceStatusEventListener (this) ;
  }

  /**
   * método definido pela interface UserEventListener.
   */
  public void UserEventReceived (UserEvent e) {

    /* tratamento do evento */

  }
}
```

Figura 34 - Exemplo do controle de eventos no DVB-J

Na Figura 34, a classe denominada *App* implementa o observador de eventos (*UserEventListener*). No construtor da classe *App* são instanciados o gerenciador de eventos, através do objeto *em*, e o repositório *R1*, através do objeto *repository*. Ao repositório é adicionado apenas o evento da tecla de confirmação (*VK_ENTER*). Quando essa tecla é pressionada, o método *UserEventReceived()* do observador é chamado, recebendo como parâmetro o objeto *e* da classe *UserEvent*. A classe *UserEvent* define métodos que caracterizam os eventos, como o método *getCode()*, que retorna o código da tecla pressionada. Esse método seria útil, por exemplo, se mais de uma tecla fosse especificada no repositório, permitindo um tratamento diferenciado de acordo com a tecla acionada pelo usuário.

A.1.2. Linguagens baseadas em HTML

Adicionalmente às linguagens baseadas em Java, são definidas linguagens baseadas em HTML para a especificação de aplicações nos padrões de TV digital. As linguagens baseadas em HTML favorecem, a princípio, a autoria, em função da simplicidade e da difusão do HTML como linguagem para formatação de documentos na Web (*World-Wide Web* ou *WWW*) (Berners-Lee et al., 1994). No entanto, é importante ressaltar que as funcionalidades do HTML são restritas,

quando é necessário realizar especificações de sincronismo e de interatividade (que não só a de relacionamentos de referência). Para superar essa limitação o HTML é adotado em conjunto com linguagens de script. Nesse caso, a complexidade da especificação torna-se próxima à complexidade da programação procedural ou orientada a objetos[10].

De forma similar às linguagens baseadas em Java, as linguagens dos padrões de TV digital baseadas em HTML utilizam um conjunto de recursos oferecidos pelos *middlewares*. Entre esses recursos, podem ser destacados os interpretadores de tecnologias relacionadas ao HTML, como o DOM (*Document Object Model*) (W3C, 2004c) e CSS (*Cascading Style Sheets*) (W3C, 2004b). Em relação às linguagens (baseadas em HTML) DVB-HTML, ACAP-X e BML/B-XML definidas pelos padrões DVB, ATSC e ISDB, respectivamente, pode haver variações quanto à versão ou a disponibilidade dos recursos no *middleware*, porém, no contexto geral, todas elas suportam os recursos já citados e, adicionalmente, uma linguagem de script.

As aplicações nas linguagens baseadas em HTML possuem um ciclo de vida similar ao ciclo das aplicações baseadas em Java. No DVB-HTML, por exemplo, podem haver várias aplicações simultâneas, controladas por uma entidade denominada agente do usuário (*user agent*), residente no *middleware* (Soares et al., 2004). A Figura 35 apresenta os cinco estados de uma aplicação DVB-HTML: carregando (*loading*), pausado (*paused*), ativo (*active*), destruído (*destroyed*) e finalizado (*killed*).

[10] As linguagens de script, como o ECMAScript, usualmente podem ser definidas através de construções típicas do paradigma procedural, ou, em alguns casos, utilizando objetos e as características do paradigma orientado a objetos.

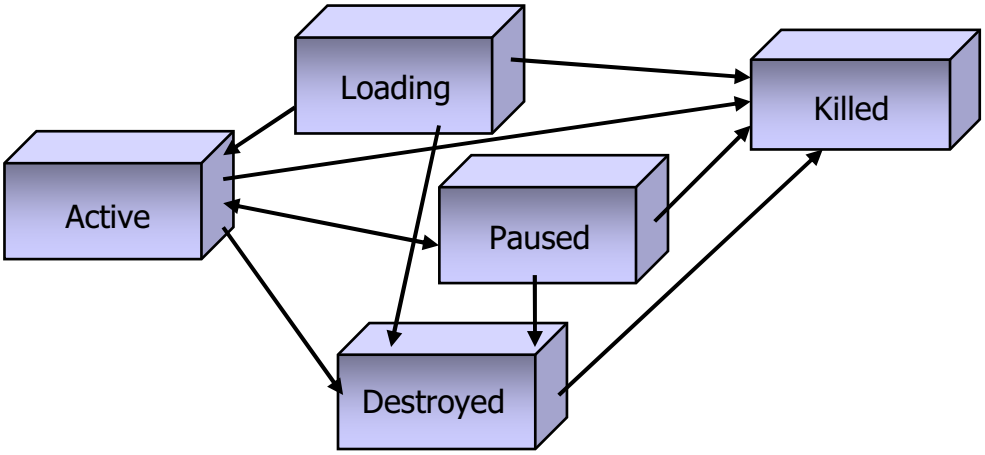

Figura 35 - Ciclo de vida de uma aplicação DVB-HTML

Quando uma aplicação DVB-HTML é recebida, ela encontra-se no estado carregando. Nesse estado, a aplicação deve realizar o acesso ao conteúdo dos objetos de mídia referenciados. Se todos os conteúdos iniciais estiverem disponíveis, a aplicação é alterada para o estado ativo. Uma vez no estado ativo, uma aplicação pode ser transferida para o estado pausado, em virtude de alguma restrição nos recursos necessários à aplicação ou ainda, devido ao fato da aplicação ter provocado alguma violação das permissões de acesso aos recursos. Uma aplicação no estado pausado pode ser reativada e voltar para o estado ativo, quando terminarem as restrições que levaram a aplicação a esse estado. Quando a aplicação é removida do sistema, por exemplo, pela perda de um recurso, define-se que ela encontra-se no estado destruído (*destroyed*). Finalmente, quando a aplicação termina sua execução, ela muda para o estado finalizado (*killed*). Nesse estado todos os recursos utilizados pela aplicação são liberados.

As linguagens baseadas em HTML utilizadas nos padrões de TV digital são, no geral, integradas às linguagens baseadas em Java do mesmo padrão. Nos padrões DVB e ATSC, todos os recursos do *middleware* disponíveis para as aplicações Java, estão disponíveis para as aplicações baseadas em HTML. O acesso às APIs (*Application Programming Interface*) dos *middlewares* é oferecido através de chamadas usando a linguagem ECMAScript.

Além do acesso às APIs Java, nos padrões DVB e ATSC, as linguagens baseadas em HTML podem exibir aplicações construídas nas linguagens DVB-J e ACAP-J, respectivamente, como se fossem mais um conteúdo sincronizado ao áudio e vídeo principal. Como exemplo, a Figura 36 apresenta um trecho de programa especificado em DVB-HTML, onde o elemento *object*, é responsável

pela exibição de aplicações especificadas em DVB-J. Nesse exemplo, o *Xlet MyApplication.class* é exibido na região delimitada pelo elemento *object* e receberá como parâmetros os valores passados por *arg_0* e *arg_1*.

```
<object type = "application/dvbj" id = "myApplication"
        codetype = "application/javatv-xlet"
        classid = "MyApplication.class"
        codebase = "myApplication/"
        height = "200" width = "150">
   <param name = "arg_0" value = "param1" />
   <param name = "arg_1" value = "param2" />
   <embed height = "200" width = "150"
          arg_0 = "param1" arg_1 = "param2">
   </embed>
</object>
```

Figura 36 - Exemplo de chamada de aplicativo DVB-J no DVB-HTML

A.1.2.1. Sincronismo e Interatividade nas linguagens baseadas em HTML

No contexto geral, as linguagens baseadas em HTML, DVB-HTML, ACAP-X e BML/B-XML, tratam os eventos de sincronismo e interatividade de forma similar às linguagens baseadas em Java, DVB-J, ACAP-J e ARIB-B23, respectivamente. Essa característica deve-se ao uso do HTML em conjunto com o ECMAScript, que possui características do paradigma orientado a objetos.

Para implementar os pontos de sincronização pertencentes a um fluxo (*stream events*), as linguagens baseadas em HTML, no geral, também utilizam observadores (*listeners*). Em ECMAScript, os observadores podem ser definidos em relação a eventos DOM. Para permitir o uso desses eventos em aplicações especificadas em HTML, os *middlewares* realizam a conversão de *stream events* para eventos DOM.

Como exemplo, no *middleware* do DVB a conversão de *stream events* para eventos DOM é realizada através das especificações contidas em um arquivo XML denominado *event factory*. Esse arquivo deve ser colocado no mesmo diretório DSM-CC do arquivo DVB-HTML, com o mesmo nome, porém com a extensão *lnk*. A Figura 37 apresenta um código DVB-HTML que define uma função a ser executada quando o evento denominado *myTriggerEvent* for recebido. Nesse exemplo, quando o arquivo DVB-HTML é carregado, a função *setupEventListeners()* é executada. Essa função apenas informa ao observador do documento, através do método *addEventListener()*, que a função *handleEvent()* deve ser executada quando o evento denominado *myTriggerEvent* for recebido.

Conforme mencionado, a associação entre o nome *myTriggerEvent* e os *stream events* que são recebidos pela aplicação é definida no arquivo *event factory*.

```
<?xml version =”1.0”?>
   <!DOCTYPE html PUBLIC “//DVB//DTD XHTML DVB-HTML 1.0//EN”
      “http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd” >
   <html xmlns = “http://www.w3.org/1999/xhtml”
         xmlns:dvbhtml = “http://www.dvb.org/mhp” >
      <head>
         <script type = ”text/ecmascript”>
            function handleEvent(evt) {
              /*código do evento*/
            }
            function setupEventListeners() {
              var htmlNode = document.documentElement;
              htmlNode.addEventListener(
                  “myTriggerEvent”, handleEvent, true);
            }
         </script>
      </head>
      <body dvbhtml:onload = “setupEventListeners()”>
      </body>
   </html>
```

Figura 37 - Tratamento de *stream events* no DVB-HTML

Nas linguagens baseadas em HTML, os eventos de interatividade podem, a princípio, ser definidos através de relacionamentos de referência, utilizando os elos HTML. Os elos são disparados por eventos de interatividade, como o pressionamento de uma tecla específica do controle remoto, e podem ter como ação a exibição de um novo documento. Outras ações também podem ser executadas, como, por exemplo, a execução de funções ECMAScript.

A Figura 38 mostra, como exemplo, um trecho de um documento DVB-HTML contendo um elo que define um relacionamento de referência para outro documento. No exemplo, o elo definido pela *tag a*, tem configurado o atributo *accesskey* com valor *&VK_GUIDE;* que equivale a ação de pressionar a tecla do guia de programação no controle remoto. No DVB-HTML, os mnemônicos relativos às teclas do controle remoto são definidos na DTD (*Document Type Definition*) dessa linguagem.

```
<?xml version =”1.0”?>
   <!DOCTYPE html PUBLIC “//DVB//DTD XHTML DVB-HTML 1.0//EN”
      “http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd” >
   <html xmlns = “http://www.w3.org/1999/xhtml”
         xmlns:dvbhtml = “http://www.dvb.org/mhp” >
      <head> </head>
      <body>
         <a accesskey=”&VK_GUIDE;” href=”guide/contents.html”>
           informações do guia de serviços
         </a>
      </body>

   </html>
```

Figura 38 - Elo de interatividade no DVB-HTML

Mais detalhes sobre linguagens nos padrões de TV digital podem ser encontrados em (Soares et al., 2004).

A.2. Linguagens Declarativas

Para superar a complexidade no desenvolvimento de aplicações para TV digital utilizando linguagens baseadas em Java ou em HTML + scripts, linguagens declarativas podem ser utilizadas. As linguagens declarativas favorecem a autoria, pois permitem representar as relações entre eventos em documentos multimídia/hipermídia sem que seja necessário especificar estruturas complexas, como estruturas de controle, repetição, variáveis, observadores etc. Em outras palavras, além de possuírem uma estrutura de marcação (XML), cada marcação tem um significado semântico que permite representar, inclusive, eventos de sincronismo e interatividade.

Atualmente, as linguagens declarativas não são encontradas, de forma efetiva, nos padrões de TV digital. No entanto, essas linguagens são amplamente difundidas para a modelagem de aplicações multimídia/hipermídia, com destaque para as linguagens SMIL (Bulterman & Rutledge, 2004) e NCL (Muchaluat-Saade et al., 2003)[11].

A linguagem SMIL (*Synchronized Multimedia Integration Language*) é uma linguagem baseada em XML, e especificada pelo W3C (W3C, 2001), que permite a descrição de apresentações multimídia interativas. Essa linguagem é bastante intuitiva para o desenvolvimento de apresentações, pois é possível definir de forma simples como os objetos de mídia estarão dispostos no tempo através de um conjunto básico de composições que possuem semântica de sincronização. São três os tipos de composições de sincronização temporal:

- Seqüencial (*seq*): os objetos de mídia são executados seqüencialmente, onde um objeto nunca começa antes que seu predecessor termine;
- Paralela (*par*): os objetos de mídia são executados em paralelo. Isso não significa que eles terminarão juntos, mas sim que esses objetos são iniciados simultaneamente;

[11] Além dessas linguagens, outras poderiam ser citadas. A linguagem declarativa do padrão MPEG-4, XMT-O [ISO, 2001] poderia ser utilizada para especificar eventos nas aplicações para TV digital. No caso específico do XMT-O, os módulos de sincronismo e interatividade são praticamente os mesmos da linguagem SMIL. Dessa forma, as definições de SMIL para TV digital também podem ser aplicadas no caso do XMT-O.

- Exclusiva (*excl*): somente um objeto de mídia que compõe esse tipo de composição pode estar ativo por vez. A ativação desses objetos é geralmente feita sob demanda, como por exemplo, ao ocorrer um evento de clique (*begin="button1.activateEvent"*).

Vale ressaltar que as composições SMIL podem ser aninhadas para elaboração de uma apresentação, ou seja, uma composição pode ser composta por objetos de mídia e/ou outras composições, e assim recursivamente.

Já a NCL (*Nested Context Language*) é uma linguagem para autoria de documentos hipermídia baseada no modelo conceitual NCM (*Nested Context Model*) (Soares et al., 2003). Para compreender as características dessa linguagem, é importante compreender as características do modelo no qual ela é baseada. No modelo NCM um documento hipermídia é representado por um nó de composição, podendo este conter um conjunto de nós que podem ser objetos de mídia ou outros nós de composição, recursivamente, e ainda elos relacionando esses nós.

Nós de composição no modelo NCM não têm nenhuma semântica embutida, diferente das composições SMIL que, por exemplo, têm semântica temporal. A semântica dos relacionamentos entre as entidades de um documento é feita através dos elos NCM, que podem especificar relações de referência, relações de sincronização, relações de derivação, relações entre tarefas de um trabalho cooperativo etc. Os elos NCM são definidos fazendo referência a um conector hipermídia, que pode representar qualquer tipo de relação. Além da referência a um conector, um elo define um conjunto de *binds* associando papéis do conector a nós do documento. Elos ainda podem representar relacionamentos multiponto entre vários nós de um documento.

A.2.1. Sincronismo e Interatividade nas linguagens declarativas

As linguagens SMIL e NCL são compostas por um conjunto de módulos que agrupam elementos com funcionalidades semelhantes. Ambas as linguagens definem um conjunto de módulos destinados a suportar eventos de sincronismo e de interatividade.

Na linguagem SMIL, as composições são definidas nos módulos *BasicTimeContainers* (*par* e *seq*) e *ExclTimeContainers* (*excl*). Além das composições com semântica de sincronização, os relacionamentos também podem ser definidos pela igualdade de eventos. Utilizando eventos de início da apresentação do nó (*begin*), término da apresentação do nó (*end*), seleção espacial (*click*) e alguns outros (Bulterman & Rutledge, 2004), é possível estabelecer relacionamentos de sincronização entre nós (objetos de mídia, por exemplo) de um documento.

O módulo *BasicInlineTiming* define os atributos *begin* e *end*, que podem ser utilizados nos elementos que representam os objetos de mídia e as composições. Nesse ponto vale destacar que os eventos de início e término de apresentação (*begin* e *end*, respectivamente) desempenham papéis muito parecidos ao de elos NCM. No módulo *BasicInlineTiming* também é definido o atributo *dur*, que especifica a duração explicita de um objeto de mídia ou de uma composição.

Em relação à interatividade, SMIL define eventos relativos às ações do mouse (*mouse up, down, out* e *over*) no módulo *SyncbaseTiming*, e do teclado no módulo *AccessKeyTime*[12].

Na linguagem NCL, os relacionamentos entre eventos são definidos através do perfil *XConnector* (Muchaluat-Saade et al., 2003), o qual permite especificar conectores definindo relações entre eventos. Existem diversos tipos de eventos que podem ser referenciados nos conectores, incluindo eventos de sincronismo e de interatividade.

Um conector especifica a relação, sem mencionar os nós que irão participar do relacionamento. Elos definem um relacionamento fazendo referência a um conector hipermídia e a um conjunto de associações, que definem os participantes do relacionamento.

A princípio, no SMIL a especificação de relacionamentos de sincronização é mais fácil, pois as composições possuem semântica de sincronização. No entanto, esse benefício é limitado pela existência de um conjunto simples de composições (seqüencial, paralela e exclusiva), e isso pode dificultar a definição de

[12] Em XMT-O, adicionalmente a SMIL, é definido o módulo *XMTEvents* com outros eventos relativos à apresentação, tais como: colisão e aproximação de objetos em animações, visibilidade etc.

relacionamentos complexos, podendo ser necessário, para esses casos, estabelecer composições com vários níveis de aninhamento, ou o uso de eventos SMIL, que podem fazer com que a composição perca sua semântica associada.

Na linguagem NCL, as composições podem herdar a semântica definida em um outro perfil da linguagem denominado *XTemplate* (Muchaluat-Saade et al., 2003). Nesse perfil, os *templates* de composição especificam tipos de componentes, tipos de relações, componentes e relacionamentos que uma composição possui ou pode possuir, sem identificar todos os componentes e relacionamentos, pois essa especificação fica sob responsabilidade da composição que utilizar o *template*. Como caso particular pode ser definida a semântica das composições SMIL.

Em relação aos *stream events* que podem ser recebidos pelo *middleware* estabelecendo pontos de sincronização em relação a um fluxo, a linguagem SMIL não define um perfil que manipule esse tipo de evento através da linguagem. De forma similar, na linguagem NCL não existem elementos nos módulos que possam formar um perfil voltado para o tratamento de *stream events*. No entanto, o uso da linguagem NCL para aplicações de TV digital prevê a recepção dos *stream events* pelo *middleware*, onde esses metadados serão interpretados para modificação da especificação da linguagem durante a exibição.

Em (Costa et al., 2006) é proposta uma arquitetura onde os *stream events* são tratados como metadados para edição do modelo que a especificação em NCL de uma aplicação qualquer representa. É relativamente comum a interpretação dos *stream events* pelo *middleware* nas linguagens dos padrões para TV digital descritas na Seção A.1. Nessas linguagens, os *stream events* são convertidos em eventos detectáveis aos observadores das linguagens, que, por sua vez, podem definir métodos para implementar as ações correspondentes aos eventos recebidos.

Na linguagem NCL, os *stream events* podem ser criados a partir de operações de edição realizadas durante a apresentação das aplicações (edição ao vivo). Entre outras vantagens, essa proposta preserva a estrutura original do documento no ambiente de execução.

A proposta apresentada em (Costa et al., 2006) evita também a necessidade de operações complexas de programação para definir métodos ou funções condicionais em relação aos *stream events* recebidos. Essa proposta integra os

ambientes de autoria e de execução, abstraindo do autor a criação ou a semântica dos *stream events*, cujo controle fica completamente a cargo do editor e do exibidor. Com isso, o autor precisa apenas se preocupar com a semântica do documento e não com a forma com que o sincronismo será implementado.

Em algumas propostas de *middleware* para SMIL, como em (Pihkala et al., 2002), (Lamdon et al., 2003) e (ICE-CREAM, 2007), não é mencionado como os *stream events* podem ser tratados. Nessas propostas, a linguagem SMIL é utilizada em conjunto com um *middleware* procedural definido nos padrões de TV digital. A Figura 39 apresenta a arquitetura proposta em (ICE-CREAM, 2007).

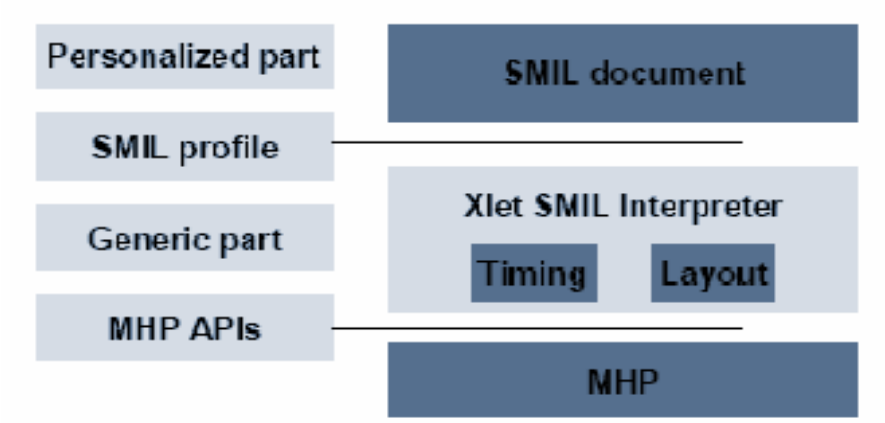

Figura 39 - Proposta de uma arquitetura SMIL para TV digital

A proposta apresentada na Figura 39 define um *Xlet* capaz de interpretar especificações de um documento SMIL. O *Xlet*, por sua vez, é implementado em Java, sobre um *middleware* MHP. O *Xlet* acessa as funções do *middleware*, através da sua API, com o objetivo de interpretar e obedecer às especificações de um perfil SMIL voltado para TV digital. O perfil em questão é similar aos perfis SMIL que podem ser normalmente definidos através da linguagem. Nesse caso, a diferença consiste em alguns módulos, como o módulo *AccessKeyTime*, da área funcional *Timing*, que tem alguns elementos estendidos a fim de capturar eventos gerados pelas teclas do controle remoto.

A Figura 40 ilustra a arquitetura proposta em (Pihkala et al., 2002) e (Lamdon et al., 2003). Na realidade, as características dessa arquitetura são similares às características apresentadas em (ICE-CREAM, 2007). Nela o gerenciador de aplicações controla *Xlets* que podem ser interpretadores SMIL ou outras aplicações quaisquer.

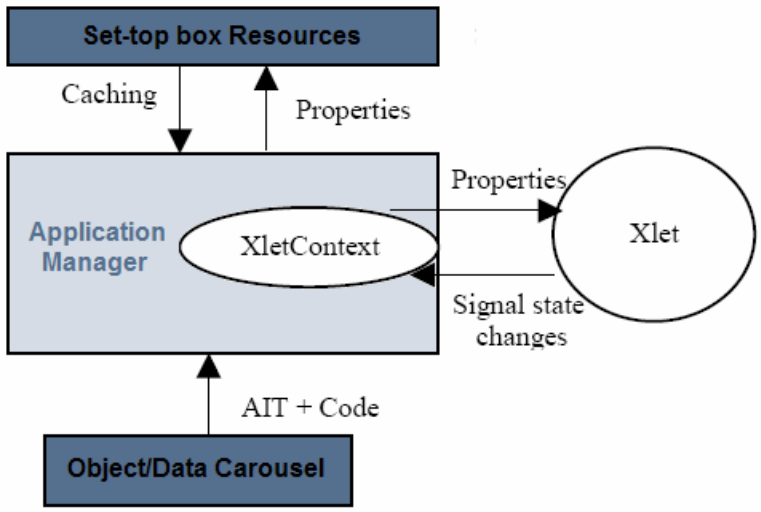

Figura 40 - Gerenciamento de *Xlets* contendo interpretadores SMIL