8
Referências

ABNT – Associação Brasileira de Normas e Técnicas. **Televisão digital terrestre – Sistema de Transmissão.** ABNT, NBR 15601:2007. Versão corrigida. 2008.

ABNT – Associação Brasileira de Normas e Técnicas. **Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações.** ABNT, NBR 15606-2:2007. Versão corrigida 3. 2009.

ALLEN, J.F.; **Maintaining Knowledge about Temporal Intervals**, Comunication of the ACM, 26. Novembro, 1983.

ARIB – Association of Radio Industries and Businesses. **Data Coding and Transmission Specifications for Digital Broadcasting**. ARIB, STD-B24, Volume 2: XML-Based Multimedia Coding Schema. 2005.

ATSC - Advanced Television Systems Committee. **ATSC Standard: Advanced Common Application Platform (ACAP)**. Padrão A/101. 2005.

BACHELET, B.; MAHEY, P.; RODRIGUES, R.F.; SOARES, L.F.G. **Elastic Time Computation in QoS-driven Hypermedia Presentation**. ACM Multimedia Systems. Maio, 2007.

BAECKER, R.; ROSENTHAL, A.J.; FRIEDLANDER, N.; SMITH, E.; COHEN, A. **Multimedia System for Authoring Motion Pictures**. ACM Multimedia. Boston, EUA. 1996.

BERTINO, E.; FERRARI, E. **Temporal Synchronization Models for Multimedia Data**. IEEE Transactions on Knowledge and Data Engineering. Agosto, 1998.

BLAKOWSKI, G.; STEINMETZ, R. **A Media Synchronization Survey: Reference Model, Specification, and Case Studies**. IEEE Journal on Selected Areas in Communications, n. 14. Janeiro, 1996.

BUCHANAN, M.C.; ZELLWEGER, P.T. **Specifying Temporal Behavior in Hypermedia Documents**. European Conference on Hypertext. Milão, Itália. Dezembro, 1992.

BUCHANAN, M.C.; ZELLWEGER, P.T. **Automatic Temporal Layout Mechanisms**. ACM Multimedia. California, EUA. Agosto, 1993.

BULTERMAN, D.C.A.; VAN ROSSUM, G.; VAN LIERE, R. **A Structure for Transportable Dynamic Multimedia Documents**. USENIX Conference. 1991.

BULTERMAN, D.C.A.; JANSEN, J.; KLEANTHOUS, K.; BLOM, K.; BENDEN, D. **AMBULANT: A Fast, Multi-Platform Open Source SMIL Player**. ACM International Conference on Multimedia. Nova York, EUA. 2004.

BULTERMAN, D.C.A.; HARDMAN, L. **Structured Multimedia Authoring**. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP). ISSN 1551-6857. Nova York, EUA. 2005.

CESAR, P.; BULTERMAN, D.C.A.; JANSEN, J. **An Architecture for End-User TV Content Enrichment**. European Interactive TV Conference. Atenas, Grécia. 2006.

CESAR, P.; BULTERMAN, D.C.A.; OBRENOVIC, Z.; DUCRET, J.; CRUZ-LARA, S. **An Architecture for Non-Intrusive User Interfaces for Interactive Digital Television Experiences**. European Interactive TV Conference. Amsterdam, Holanda. 2007.

CESAR, P.; BULTERMAN, D.C.A.; GEERTS, D.; JANSEN, J.; KNOCHE, H.; SEAGER, W. **Enhancing Social Sharing of Videos: Fragment, Annotate, Enrich, and Share**. ACM International Conference on Multimedia. Vancouver, Canadá. Outubro, 2008.

CHUNG, S.M.; PEREIRA, A.L. **Timed Petri Net Representation of SMIL**. IEEE Multimedia, v. 12, n. 1. Março, 2005.

COSTA, R.M.R; MORENO, M.F.; RODRIGUES, R.F.; Soares L.F.G. **Live Editing of Hypermedia Documents.** ACM Symposium on Document Engineering. Amsterdam, Holanda. 2006.

COSTA R.M.R; MORENO M.F.; SOARES L.F.G. **Intermedia Synchronization Management in DTV Systems.** ACM Symposium on Document Engineering. São Paulo, Brasil. 2008.

COSTA, R.M.R; MORENO, M.F.; SOARES, L.F.G. **Ginga-NCL: Suporte a Múltiplos Dispositivos.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Natal, Brasil. 2009.

CONCOLATO, C.; LE FEUVRE, J.; MOISSINAC, J.C. **Timed-Fragmentation of SVG Documents to Control the Playback Memory Usage**. ACM Symposium on Document Engineering. Nova York, EUA. 2007.

DINI, R.; PATERNO, F.; SANTORO, C. **An Environment to Support Multi-User Interaction and Cooperation for Improving Museum Visits through Games**. Mobile Human-Computer Interaction Conference. Cingapura. Setembro, 2007.

ETSI – European Telecommunications Standards Institute. **Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1.** Especificação Técnica ETSI TS 102 B12. 2005.

HERPEL, C. **Elementary Stream Management in MPEG-4**. IEEE Transactions on Circuits and Systems for Video Technology, v. 9, n. 2. Março, 1999.

GAREY, M.R.; JOHNSON, D.S. **Computers and Intractablitiy: A Guide to the Theory of NP-Completeness**. ISSN 978-0716710455. W. H. Freeman. Nova York, EUA. 1979.

ISO/IEC - International Organization for Standardization 13818-6. **Information technology – Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC**. 1998.

ISO/IEC - International Organisation for Standardisation. **14496-5:2000. Coding of Audio-Visual Objects – Part 5: Reference Software.** 2ª Edição. 2000.

ISO/IEC - International Organisation for Standardisation. **13818-1:2000. Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems.** 2000.

ISO/IEC - International Organisation for Standardisation. **14496-1:2001. Coding of Audio-Visual Objects – Part 1: Systems.** 2ª Edição. 2001.

ISO/IEC - International Organization for Standardization. **14496-20:2006. Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format.** 2006.

ITU-T – International Telecommunication Union - **Recommendation H.761 - Nested Context Language (NCL) and Ginga-NCL for IPTV Services**. Geneva, Suição. Abril, 2009.

JOURDAN, M.; ROISIN, C.; TARDIF, L. **Madeus, an Authoring Environment for Interactive Multimedia Documents**. ACM Multimedia Conference. Inglaterra. Setembro, 1998.

KIM, M.; WOOD, S. **MPEG-4 Flexible Timing Standard (FlexTime)**. Overview of FlexTime. IBM Research. 2002. Disponível em http://www.research.ibm.com/mpeg4/Projects/Flextime.htm. Acesso em: 05 jul. 2010.

LITTLE, T.; GHAFOOR, A. **Synchronization and Storage Models for Multimedia Objects**, IEEE Journal on Selected Areas in Communications, v. 8, n. 3. ISSN 0733-8716. Abril, 1990.

MORENO, M.F.; COSTA, R.M.R; SOARES, L.F.G. **Sincronismo entre Fluxos de Mídia Contínua e Aplicações Multimídia em Redes por Difusão.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Vilha Velha, Brasil. 2008.

MORENO, M.F. **Ginga-NCL: Relating Imperative, Declarative and Media Objects.** Workshop of Thesis and Dissertations. European Interactive TV Conference. Leven, Bélgica. 2009.

MORENO, M.F.; SOARES, L.F.G; CERQUEIRA, R. **Uma Arquitetura Orientada a Componentes para o Middleware Ginga-NCL.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Belo Horizonte, Brasil, 2010.

MUCHALUAT-SAADE, D.C.; RODRIGUES, R.F.; SOARES, L.F.G. **XConnector: Extending XLink to Provide Multimedia Synchronization.** ACM Symposium on Document Engineering. McLean, EUA. 2002.

NETO, C.S.S; SOARES, L.F.G. **Reuso e Importação em Nested Context Language**. Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Fortaleza, Brasil. 2009.

PATERNO, F.; SANTORO, C.; MANTYJARVI, J.; MORI, G.; SANSONE, S. **Authoring Pervasive Multimodal User Interfaces.** International Journal of Web Engineering and Technology, v. 4. ISSN: 1476-1289. 2008.

PÉREZ-LUQUE, M.J.; LITTLE, T.D.C. **A Temporal Reference Framework for Multimedia Synchronization**. IEEE Journal on Selected Areas in Communications. ISSN 0733-8716. Janeiro, 1996.

PESSOA, B.J.S.; SOUZA, G.L.S.; CABRAL, L.A.F. **Metaheurísticas Aplicadas à Geração de Carrossel no Sistema Brasileiro de TV Digital.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Vilha Velha, Brasil. 2008.

RODRIGUES, R. F. **Formatação e Controle de Apresentações Hipermídia com Mecanismos de Adaptação Temporal**. 2003. 170 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 2003.

RODRIGUES, R. F.; SOAREZ, L.F.G. **Inter and Intra Media-Object QoS Provisioning in Adaptive Formatters**. ACM Symposium on Document Engineering. Grenoble, França, 2003.

SCHROYEN, J.; GABRIELS, K.; LUYTEN, K.; TEUNKENS, D.; et al. **Training Social Learning Skills by Collaborative Mobile Gaming in Museums**. International Conference on Advances in Computer Entertainment Technology. Yokohama, Japão. Dezembro, 2008.

SOARES, L.F.G; RODRIGUES, R.F.; MUCHALUAT-SAADE, D.C. **Modeling, Authoring and Formatting Hypermedia Documents in the HyperProp System**. Multimedia Systems, v. 8, n. 2. Alemanha. 2000.

SOARES, L.F.G.; RODRIGUES R.F. **Nested Context Model 3.0 Part 1 – NCM Core.** Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 18/05. ISSN 0103-9741. Rio de Janeiro, Brasil. Maio, 2005.

SOARES, L.F.G.; et al. **Sistema Brasileiro de Televisão Digital – Recomendações para o Modelo de Referência – Sincronismo de Mídias.** Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 41/05. ISSN 0103-9741. Rio de Janeiro, Brasil. Dezembro, 2005.

SOARES, L.F.G; RODRIGUES, R.F. **Nested Context Language 3.0 Part 8 – NCL Digital TV Profiles**. Relatório Técnico do Departamento de Informática da PUC-RIO, MCC 35/06. ISSN 0103-9741. Rio de Janeiro, Brasil. Outubro, 2006.

SOARES, L.F.G; RODRIGUES, R.F.; COSTA, R.M.R. **Geração Automática de Frameworks para Processamento de Documentos XML.** Simpósio Brasileiro de Sistemas Multimedia e Web – WebMedia. Natal, Rio Grande do Norte. 2006.

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. ISSN 0104-6500. 2007.

SOARES, L.F.G.; COSTA, R.M.R; MORENO, M.F.; MORENO, M.F. **Multiple Exhibition Devices in DTV Systems.** ACM International Conference on Multimedia. Beijing, China. 2009.

SOARES, L.F.G.; BARBOSA, S.D.J. **Programando em NCL 3.0 – Desenvolvimento de Aplicações para o Middleware Ginga, TV Digital e Web**. ISBN 978-85-352-3457-2. Elsevier. 2009.

SOARES, L.F.G.; MORENO, M.F.; SOARES NETO, C.S.; MORENO, M.F. **Ginga-NCL: Declarative Middleware for Multimedia IPTV Services**. IEEE Communications Magazine, v. 48, n. 6. ISSN: 0163-6804. Junho, 2010.

SOARES, L.F.G.; RODRIGUES, R.F; CERQUEIRA, R.; BARBOSA, S.D.J. **Variable and State Handling in NCL**. Multimedia Tools and Applications. ISSN: 1380-7501. Março, 2010.

SOUZA, G.L. **Sincronismo na Modelagem e Execução de Apresentações de Documentos Multimídia**. 2003. 231 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro. 1997.

SOUZA, G.L.; Leite, L.E.C; Batista, C.E.C.F. **Ginga-J: The Procedural Middleware for the Brazilian Digital TV System**. Journal of the Brazilian Computer Society, v. 13, n. 4. ISSN: 0104-6500. 2007.

VAN ROSSUM, G.; JANSEN, J.; MULLENDER, K.S.; BULTERMAN, D.C.A. **CMIFed: A Presentation Environment for Portable Hypermedia Documents**. ACM Multimedia. Anaheim, EUA. Agosto, 1993.

VUONG, S.; COOPER, K.; ITO, M. **Petri Net Models for Describing Multimedia Synchronization Requirements**, International Conference on Network Protocols. Japão. Novembro, 1995.

W3C - World-Wide Web Consortium. **XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)**. W3C Recommendation. Agosto, 2002.

W3C - World-Wide Web Consortium. **Scalable Vector Graphics – SVG 1.1 Specification**. W3C Recommendation. Janeiro, 2003.

W3C - World-Wide Web Consortium. **XML Schema Part 0: Primer (Second Edition)**. W3C Recommendation. Outubro, 2004.

W3C - World-Wide Web Consortium. **XML Schema Part 1: Structures (Second Edition)**. W3C Recommendation. Outubro, 2004.

W3C - World-Wide Web Consortium. **XML Schema Part 2: Datatypes (Second Edition)**. W3C Recommendation. Outubro, 2004.

W3C - World-Wide Web Consortium. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. W3C Recommendation. Novembro, 2008.

W3C - World-Wide Web Consortium. **Synchronized Multimedia Integration Language (SMIL 3.0) Specification**. W3C Recommendation. Dezembro, 2008.

WOO, M.; QAZI, N.; GHAFOOR, A. **A Synchronization Framework for Communication of Pre-Orchestrated Multimedia Information**. IEEE Network. Fevereiro, 1994.

YAMADA, F.; BEDICKS, G.; CASTRO, P.H. **Modulação do Sistema Brasileiro de TV Digital**. Revista Produção Profissional – A Modulação do SBTVD. Edição 74. Fevereiro, 2008.

YANG, C.C.; HUANG, J.H. **A Multimedia Synchronization Model and Its Implementation in Transport Protocols**, IEEE Journal on Selected Areas in Communications, v. 14, n.1. Janeiro, 1996.

YANG, C.C. **Detection of the Time Conflicts for SMIL-Based Multimedia Presentations**. Workshop on Computer Networks, Internet and Multimedia. International Computer Symposium. Taiwan, 2000.

YANG, C.C.; TIEN, C.W.; WANG, Y.C. **Modeling of the Non-Deterministic Synchronization Behaviors in SMIL 2.0 Documents**, International Conference on Multimedia and Expo, v. 3. Julho, 2003.

Apêndice A
Sincronismo Temporal de Aplicações Hipermídia

Nas aplicações hipermídia, a especificação de quando os eventos associados às mídias deverão ocorrer corresponde ao seu sincronismo temporal. Essas aplicações podem conter mídias com conteúdo dinâmico, como vídeos, áudios e animações e também mídias com eventos associados entre si, como na exibição simultânea de um áudio e um vídeo com conteúdos semanticamente relacionados. Esse último caso, que constitui o foco deste capítulo, é usualmente denominado sincronismo intermídia, sendo diferente do primeiro caso, usualmente denominado sincronismo intramídia, que trata da sincronização entre as várias unidades que compõem um mesmo objeto.[24]

Diferentes modelos podem ser utilizados para controlar o sincronismo temporal das aplicações. Em (Blakowski, 1996) esses modelos são classificados em eixo do tempo, baseados em intervalos, hierárquicos, scripts, baseados em redes de Petri, baseados em pontos de referência e baseados em eventos. Essa, no entanto, não é a única classificação existente. Em (Bertino, 1998), por exemplo, os modelos são classificados em grafos, redes de Petri, orientados a objetos e baseados em linguagens.

Nas classificações propostas em (Blakowski, 1996) e (Bertino, 1998), os modelos em comum são descritos, basicamente, com as mesmas características. Diferentes classificações não implicam divergências em relação às características dos modelos considerados, elas normalmente são o resultado de diferentes objetivos de avaliação. Em (Bulterman, 2005), por exemplo, onde a análise dos modelos para o sincronismo das aplicações é voltada à autoria, os principais modelos são classificados em eixo do tempo, estruturados, scripts e grafos.

Neste capítulo não é adotada uma classificação específica. O foco são os modelos e, assim, foram eles os escolhidos e não suas possíveis classificações. Porém, a fim de organizar as comparações entre os modelos, quando necessário,

[24] Embora distintos, os tipos de sincronização devem funcionar de forma colaborativa. Para manter a sincronização intermídia, por exemplo, é importante preservar e controlar a sincronização intramídia dos objetos relacionados (Rodrigues, 2003b).

as classificações propostas em (Blakowski, 1996) e (Bertino, 1998) foram utilizadas. Por fim, cabe mencionar que este capítulo não tem por objetivo esgotar a descrição de todos os modelos existentes e sim priorizar aqueles utilizados em padrões e ferramentas voltados à apresentação das aplicações.

A.1. Modelos de Sincronização Temporal

A.1.1. Sincronização Baseada em um Eixo do Tempo (*Timeline*)

Existem várias alternativas para a construção de estruturas baseadas em eixo do tempo, das quais podem ser destacadas a multiplexação síncrona, o selo de tempo (*timestamping*) e a linha de tempo (*timeline*) (Soares, 2005b); sendo essa última utilizada tanto para a especificação quanto para a apresentação das aplicações (Bulterman, 2005), o que contribuiu para que a expressão *timeline* fosse utilizada em equivalência à de eixo do tempo.

A sincronização *timeline* tem por base a associação de um tempo absoluto a cada evento. Essa associação é realizada em relação a uma base, por exemplo, o início da apresentação de uma aplicação, assim, os eventos são posicionados em um eixo abstrato do tempo, com origem, por exemplo, no início da apresentação.

Na autoria, a sincronização *timeline* oferece uma abstração do comportamento temporal de um documento (Bulterman, 2005), através da qual o autor pode conhecer a distribuição das mídias no tempo sem conhecer detalhes sobre a implementação do seu sincronismo. No entanto, vários são os problemas conhecidos desse modelo (Bulterman, 2005; Soares, 2005b). Através dele não é possível especificar eventos cujo início ou duração não podem ser previstos antes do início da apresentação, como, por exemplo, as interações dos usuários. Esse modelo também pode tornar difícil a manutenção das aplicações, uma vez que modificações no posicionamento dos eventos associados a uma mídia podem exigir modificações nos momentos temporais associados à execução de outros eventos, a fim de preservar a semântica original da aplicação.

Para superar as limitações da sincronização *timeline*, uma estratégia possível é utilizar construções baseadas nesse modelo apenas para a apresentação e transmissão das aplicações, utilizando um outro modelo na autoria. Essa opção é,

por exemplo, oferecida pelo MPEG-4 Sistemas (ISO/IEC, 2001), que define um formato binário denominado BIFS (*Binary Format for Scenes*) (ISO/IEC, 2001) para a apresentação e o transporte das aplicações através da sincronização *timeline*.[25]

Especificações BIFS são projetadas com o objetivo de serem multiplexadas de forma síncrona em fluxos que possam ser apresentados tão logo sejam recebidos pelos clientes. BIFS permite que as modificações realizadas nas aplicações sejam atualizadas nos clientes simultaneamente à apresentação. Cada apresentação BIFS é organizada em cenas, cada uma estruturada através de uma hierarquia de objetos que representam os conteúdos das mídias. Os conteúdos das mídias em uma cena podem ser atualizados durante a apresentação e as próprias cenas também podem ser atualizadas através da inserção ou exclusão de objetos ou ainda através de mudanças nas propriedades dos objetos. Também é possível inserir ou excluir eventos relacionados aos objetos em cena, como eventos de animação e interação. É possível ainda substituir uma cena inteiramente por outra (Herpel, 1999).

Apesar de adotar a sincronização timeline, BIFS permite que sejam especificadas aplicações com alguma adaptação e interação. Múltiplos conteúdos relacionados a uma mesma mídia da aplicação podem ser simultaneamente transmitidos, para que um desses conteúdos seja escolhido durante a apresentação. No entanto, caso os conteúdos a serem adaptados tenham durações distintas, não será possível associar o evento de apresentação do conteúdo escolhido a outras mídias da aplicação.

Em relação à interatividade, BIFS define dois casos particulares. Um deles consiste na possibilidade de alterar propriedades relacionadas à apresentação das mídias como, por exemplo, definir a exibição de uma mídia visível (verdadeiro) ou não (falso), como resultado de uma ação interativa. Para mídias como imagens e textos, essa possibilidade permite simular que as mídias são apresentadas em função da ocorrência de um evento interativo, quando, na verdade, as mídias já estavam sendo executadas, porém de forma invisível. No entanto, para mídias que possuem durações associadas, como áudios e vídeos, essa simulação não é

[25] BIFS opcionalmente oferece suporte a algumas construções temporais usando um modelo chamado *FlexTime* (Kim, 2002) que implementa um conjunto limitado das relações de Allen (Allen, 1983). Essa opção, no entanto, não faz parte da implementação de referência para apresentações desse formato (ISO/IEC, 2000a).

possível. Outra possibilidade, relacionada à interatividade, consiste em substituir uma cena inteiramente por outra, a partir da ocorrência de um evento interativo, o que corresponde a encerrar a apresentação atual e iniciar a apresentação de uma outra aplicação.

Uma aplicação original pode ser dividida em um conjunto de pequenas aplicações, cada uma formada por um conjunto de mídias que possuem eventos associados a um mesmo momento temporal. Através dessa abordagem, eventos interativos podem ser utilizados para encerrar a apresentação atual e iniciar a apresentação de uma outra aplicação, ambas obtidas a partir da aplicação original. Essa estratégia, além de definida no MPEG-4, também pode ser utilizada nos sistemas de TV digital europeu (ETSI, 2005), americano (ATSC, 2005) ou japonês (ARIB, 2005), onde linguagens baseadas em XHTML (W3C, 2002) são oferecidas para autoria declarativa. Nesse caso, várias aplicações XHTML, cada uma com a especificação do leiaute espacial de um conjunto de mídias, podem ter suas apresentações iniciadas a partir de eventos interativos disparados por outras aplicações ou ainda através de eventos de sincronismo (ISO/IEC, 1998), transmitidos para os clientes em momentos específicos da apresentação do conteúdo audiovisual principal.

Dividir uma aplicação em várias pequenas aplicações é conveniente apenas para aplicações simples e, mesmo nesse caso, a semântica original pode ser completamente perdida. Em seu lugar tem-se um conjunto de apresentações isoladas executadas no tempo. A semântica da aplicação original permanece apenas na idéia do autor, que passa a ser também o controlador da apresentação, tarefa que para aplicações com um grande número de mídias e eventos pode ser difícil.

Quando utilizado na apresentação e transporte das aplicações, o sincronismo *timeline* não é capaz de representar os relacionamentos entre os eventos especificados na autoria. Mesmo as facilidades propostas pelo MPEG-4 para o formato BIFS não impedem que a semântica dos relacionamentos entre os eventos especificados de forma relativa na autoria seja perdida na apresentação, restando apenas os momentos previstos para o disparo dos eventos.

A.1.2.
Sincronização Hierárquica

Na sincronização hierárquica a especificação do sincronismo é realizada baseada em duas operações principais: a sincronização paralela e serial de ações (Baecker, 1996). Uma ação pode ser atômica ou composta. Ações atômicas estão relacionadas com a apresentação de uma única mídia ou com uma interação do usuário. Ações compostas são formadas por um conjunto de ações atômicas e operações de sincronização (paralela e sequencial). Graficamente, essa forma de sincronização pode ser definida como uma árvore, onde os nós folhas (terminais) representam as ações atômicas e os demais as operações de sincronização.

É possível que existam variações na estrutura proposta para que, por exemplo, seja possível especificar atrasos associados às ações. Um atraso pode ser especificado, por exemplo, como uma ação atômica, sendo aplicado as demais ações dentro de uma mesma ação composta (vanRossum, 1993).

Na Figura 9.1, uma árvore representando a sincronização hierárquica de uma aplicação formada por dois vídeos, dois áudios e dois conteúdos textuais é apresentada. Nessa aplicação, um vídeo, um áudio e um texto semanticamente relacionados são apresentados em paralelo, em sequência são apresentados, novamente em paralelo, as demais mídias.

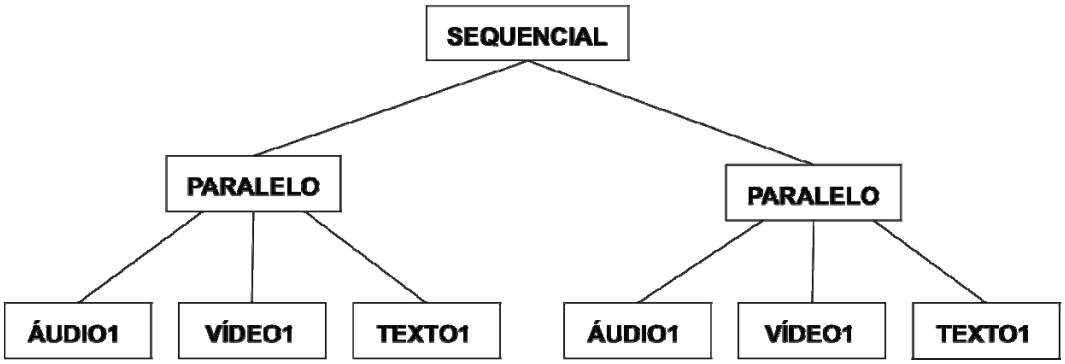

Figura A.1 - Exemplo de sincronização hierárquica.

Na sincronização hierárquica, relacionamentos entre os eventos são especificados de forma simples, através de combinações com semântica paralela e sequencial. No entanto, essa sincronização também limita a especificação dos relacionamentos apenas ao início ou ao final dos eventos de apresentação das mídias. Outros pontos de sincronização, relacionados a momentos da duração da

apresentação das mídias também podem ocorrer na sincronização hierárquica, porém de forma implícita, através da especificação de atrasos nas aplicações.

Como não é possível especificar explicitamente relacionamentos de sincronização entre os eventos que ocorrem durante a exibição de uma determinada mídia, o autor pode ser obrigado a dividir as mídias originais em várias outras. No exemplo apresentado na Figura 2, o áudio e o vídeo que são apresentados em paralelo foram divididos em duas partes, para que legendas, formadas por comentários textuais, pudessem estar sincronamente relacionadas a partes específicas (momentos temporais da duração) do vídeo. Uma outra opção seria a especificação de atrasos, mas, nesse caso, a especificação do sincronismo seria definida de forma implícita (Souza, 1997).

O formato CMIF (*CWI Multimedia Interchange Format*) (Bulterman, 1991), que corresponde a base da especificação da linguagem SMIL (*Synchronized Multimedia Integration Language*) (W3C, 2008b), recomendação W3C para a descrição de aplicações multimídia na Web, oferece construções baseadas no modelo hierárquico de sincronização. As aplicações CMIF são especificadas através de nós representando operações de sincronização paralela ou sequencial, que por sua vez podem conter ações atômicas associadas à apresentação das mídias.

O formato CMIF, além de possuir as características da sincronização hierárquica, permite que os atrasos em uma aplicação sejam especificados de forma explítica, através de relacionamentos entre ações atômicas definidas em uma mesma ação composta. Esses relacionamentos, denominados arcos de sincronização (Bulterman, 1991; vanRossum, 1993), são representados na árvore hierárquica através de arestas dirigidas entre os nós das ações atômicas envolvidas.

Na especificação CMIF do exemplo apresentado na Figura 9.1, o áudio e o vídeo são representados cada um por uma única ação, ambos sincronizados em uma mesma ação composta com semântica paralela, contendo também os dois textos correspondentes às legendas. Dois arcos de sincronização são necessários nessa aplicação, relacionando o início da exibição do vídeo com o início da exibição de cada legenda, cada um com a especificação do atraso necessário para posicionar a exibição da legenda no momento correto de exibição do vídeo.

Arcos de sincronização estendem as facilidades da sincronização hierárquica para a especificação das aplicações (vanRossum, 1993), mas não trazem, assim como os nós correspondentes às ações atômicas e operações de sincronização, facilidades à apresentação. Outros modelos de sincronização são normalmente utilizados para controlar a apresentação das aplicações especificadas de forma hierárquica. Esse é o caso, por exemplo, do exibidor do ambiente CMIFed (vanRossum, 1993), proposto para executar a apresentação de aplicações CMIF, que projeta, durante a compilação das aplicações, um grafo de dependências temporais a partir da sua especificação de sincronização hierárquica.

No grafo de dependências temporais do CMIFed, nós correspondem aos pontos de início e fim das ações compostas e atômicas e as arestas representam as relações temporais entre dois desses nós. A estrutura adotada pelo CMIFed, embora calculada a partir de uma estrutura hierárquica, possui características distintas desse modelo de sincronização, que serão discutidas ainda neste capítulo.

A.1.3. Sincronização Baseada em Redes de Petri

A estrutura de uma rede de Petri (Chung, 2005; Little, 1990; Vuong, 1995; Woo, 1994; Yang, 2003) pode ser utilizada para representar o sincronismo das aplicações e permitir que os momentos temporais relacionados à execução dos eventos sejam calculados (Yang, 1996; Yang, 2000; Yang 2003).

Os elementos que formam uma rede de Petri podem ser graficamente representados por um grafo dirigido, com dois tipos de nós: círculos representando os lugares (*places*) e barras representando as transições (*transitions*). No grafo, arestas dirigidas unem os círculos às barras e vice-versa. Uma aresta com origem em um círculo e destino em uma barra define o círculo de origem como uma função de entrada da transição. De maneira similar, uma função de saída é definida quando existe uma aresta com origem em uma transição e destino em um círculo. Além dessas entidades, fichas (*tokens*) podem estar associadas aos lugares existentes em uma rede. No grafo, fichas são representadas por pontos preenchidos desenhados nos círculos. As fichas definem o estado de execução de uma rede de Petri (Vuong, 1995).

As entidades tradicionais de uma rede de Petri podem ter suas definições estendidas, ou novas entidades, além daquelas citadas no parágrafo anterior,

podem ser adicionadas. Esse é o caso das *Timed Petri Nets* – TPN (Chung, 2005), onde estruturas temporais de duração (*clock*) são adicionadas aos elementos das redes de Petri. A OCPN (*Object Composition Petri Net*) (Little, 1990) é um exemplo de uma TPN onde, após receber uma ficha, um lugar passa para o estado ativo e permanece nesse estado durante o intervalo especificado pela sua duração (tempo de exibição da mídia). Uma vez no estado ativo, o lugar permanece com a ficha, que somente será liberada quando a sua duração for atingida. As transições na OCPN representam pontos de sincronização, disparados quando todos os seus lugares de origem contêm fichas liberadas. Quando uma transição é executada, as fichas passam dos lugares de origem para os lugares de destino de uma determinada transição.

Em (Vuong, 1995) são comparadas diferentes propostas de redes de Petri com enfoque temporal, que incluem a OCPN, a XOCPN (*Extended Object Composition Petri Net*) (Woo, 1994), entre outras. Infelizmente, mesmo nas propostas destinadas ao controle temporal, como nas variações das TPN, não é possível a especificação da sincronização em relação a eventos que ocorrem durante a exibição das mídias. É possível, de forma similar à sincronização hierárquica, realizar a divisão das mídias originais em partes temporalmente relacionadas. Essa, no entanto, é uma solução que torna a representação das redes de Petri mais complexa, tanto para a especificação quanto para o controle da apresentação das aplicações.

Considerando a importância das redes de Petri na modelagem de sistemas concorrentes, principalmente em relação à análise de propriedades que incluem a simulação do comportamento de uma aplicação, alguns trabalhos, que têm como objetivo facilitar a especificação de aplicações multimídia e hipermídia através de redes de Petri, merecem ser considerados.

Em (Chung, 2005) é proposta uma abordagem onde os lugares (*places*), além de obedecerem a duração de exibição das mídias, como normalmente ocorre nas TPN, podem representar intervalos temporais quaisquer ou até mesmo eventos, que podem ser, inclusive, disparados por ações interativas. Nessa proposta, cada lugar pode ter até duas fichas, de tal forma que a combinação dessas fichas define se o lugar encontra-se no estado ativo, inativo e também se a transição de destino desse lugar deve ser executada (Chung, 2005). Quando essa proposta é utilizada para o controle da apresentação, um lugar ativo representa o

início da exibição de uma mídia associada. Quando um lugar torna-se inativo, ocorre uma pausa na exibição dessa mídia, que terá sua exibição resumida somente quando o lugar em questão se tornar novamente ativo. A exibição da mídia termina quando a transição de destino do lugar considerado for executada.

A ERTSM (*Extended Real-Time Synchronization Model*) (Yang, 2003) é outro exemplo de uma rede de Petri proposta para a modelagem de aplicações hipermídia. Nessa estrutura, os lugares são classificados em lugares regulares (*regular*) ou lugares a serem executados (*enforced*). Lugares regulares têm a mesma semântica definida na OCPN, enquanto os lugares a serem executados provocam a execução imediata das transições de destino quando a ficha desses lugares encontra-se desbloqueada. Lugares a serem executados, além da exibição das mídias, podem estar associados a intervalos temporais ou a diferentes tipos de eventos, incluindo os imprevisíveis, como aqueles executados por ações interativas. O ERTSM especifica também um conjunto de novas entidades, denominadas controladores, para modelar o reinício, a repetição e o tempo mínimo dos eventos associados aos lugares. Para auxiliar a apresentação das aplicações, em (Yang, 2003) é proposto um algoritmo para o cálculo do tempo de apresentação de aplicações SMIL modeladas através do ERTSM.

Propostas como a ERTSM e a descrita em (Chung, 2005) adicionam novas entidades e opções semânticas às redes de Petri com o objetivo de facilitar a especificação das aplicações hipermídia. Em relação à apresentação, essas propostas oferecem facilidades como, por exemplo, o cálculo do tempo de apresentação das aplicações (Yang, 2003) e a detecção de conflitos temporais (Yang, 2000), incluindo a localização de relacionamentos que, embora estejam especificados, nunca irão ocorrer durante a apresentação. Por outro lado, embora estruturas baseadas em redes de Petri possuam a formalização necessária para a verificação de muitas propriedades temporais, essa análise formal não é normalmente o foco do controle da apresentação das aplicações hipermídia, e sua implementação em tempo de apresentação das aplicações pode ser difícil. Isso é particularmente verdade considerando os diversos novos elementos e as novas opções semânticas que fazem parte de muitas das variações de redes de Petri apresentadas, incluindo a própria ERTSM, a proposta em (Chung, 2005) e diversas outras (Little, 1990; Vuong, 1995; Woo, 1994; Yang, 1996).

A.1.4.
Sincronização Baseada em Causalidade/Restrição de Eventos

A sincronização baseada em eventos é aquela onde as ações associadas à apresentação, como o seu início e fim, são executadas a partir da ocorrência de eventos de sincronização, como a exibição de uma parte temporal do conteúdo de uma mídia. Nesse tipo de sincronização, a especificação de uma aplicação pode ser completamente definida de forma relativa, sem qualquer menção explícita ao tempo (Soares, 2005b).

Relacionamentos entre os eventos podem envolver causalidade ou restrição. Relacionamentos causais envolvem ações que devem ser executadas quando condições estabelecidas forem satisfeitas. Por exemplo, em relação a duas mídias simultaneamente exibidas, o fim da exibição de uma dessas mídias pode ser definido como condição para o término da outra. Por outro lado, relacionamentos de restrição estabelecem regras que devem ser obedecidas, como exemplo, duas mídias, quando simultaneamente exibidas devem terminar ao mesmo tempo. Diferente do primeiro exemplo, em que uma das mídias pode ter sua exibição encerrada antes do seu fim natural, para que a restrição do segundo exemplo seja obedecida, as últimas amostras de ambas as mídias devem ser simultaneamente exibidas.

Em relação aos eventos, além da apresentação, outras ocorrências relativas às aplicações também merecem ser consideradas, como, por exemplo, a seleção de uma mídia ou de uma parte do seu conteúdo, a atribuição de valores às propriedades das mídias ou da aplicação, entre outras. Todas essas ocorrências, independente do fato de terem uma duração ou serem atômicas, definem eventos que podem ser qualificados como condição ou ação em um relacionamento causal ou ainda fazer parte de uma restrição associada a um relacionamento.

A especificação do sincronismo baseada em relacionamentos entre eventos, com semântica causal ou de restrição, é útil para aplicações onde o sincronismo depende da ocorrência de eventos com duração variável ou mesmo imprevisível no momento da especificação. Por outro lado, uma vez que os momentos temporais absolutos para a execução dos eventos não são conhecidos, a apresentação das aplicações pode se tornar uma tarefa difícil.

É possível realizar a apresentação das aplicações sem que os momentos temporais absolutos para a execução dos eventos sejam conhecidos. Essa é, por exemplo, a proposta do sistema HyperProp (Soares, 2000), cujo modelo de execução implementa, através do paradigma de orientação a objetos (Rodrigues, 2003a), a sincronização baseada em causalidade e restrição de eventos. Nesse sistema, os conteúdos das mídias e os seus eventos associados são representados por objetos, que encapsulam suas propriedades e ações. A visibilidade entre os objetos é limitada. Dependendo dos relacionamentos existentes na aplicação, um objeto deve se registrar em outro para receber notificações a respeito de alterações nas propriedades ou ações desse outro objeto. Quando uma notificação é recebida, ações podem ser executadas no objeto registrado, gerando notificações em outros objetos e, consequentemente, executando novas ações, sucessivamente.

Quando o controle da apresentação é realizado obedecendo diretamente a especificação relativa entre os eventos, a semântica dos relacionamentos especificados na autoria é preservada na apresentação. Por outro lado, nenhuma facilidade para o controle temporal das aplicações é oferecida e, nesse caso, pode ser difícil oferecer funcionalidades associadas à apresentação, mesmo as mais simples, como, por exemplo, controlar se o final de uma apresentação foi atingido.

A.1.5. Sincronização Baseada em Grafos Temporais

Diferentes modelos de sincronização temporal baseados em grafos podem ser encontrados (Bertion, 1998; Buchanan, 1992; Jourdan, 1998, vanRossum, 1993). Alguns desses modelos são bastante especializados em relação às suas entidades, seu formato ou mesmo em relação a semântica do sincronismo que eles representam. Nesse caso, usualmente, esses modelos possuem sua própria classificação, como é o caso, por exemplo, das árvores utilizadas no modelo hierárquico e das redes de Petri.

O CMIFed utiliza para o controle da apresentação das aplicações uma estrutura denominada grafo de dependências temporais (vanRossum, 1993), contruída a partir da estrutura hierárquica de uma aplicação CMIF. Nesse grafo, os nós representam o início e o fim dos eventos e as arestas as restrições temporais dos relacionamentos entre dois nós. As arestas são obtidas a partir da estrutura de

sincronização hierárquica. Por exemplo, em uma operação de sincronização sequencial, o final de cada filho deve preceder o início do próximo filho existindo, portanto, uma aresta entre essas ações. Arestas também são utilizadas para representar a duração dos eventos e os arcos de sincronização. Quando partes específicas da apresentação das mídias são relacionadas através dos arcos de sincronização, nós específicos são construídos. Esses nós, por sua vez, são temporalmente relacionados, através de outras arestas, a outras partes da mesma mídia.

Dois nós especiais são construídos no grafo de dependências temporais, um representando o início e outro o fim da apresentação de uma aplicação. Uma aresta sem restrições temporais é utilizada para unir o nó de início da apresentação ao nó que representa a raiz da estrutura hierárquica. Os nós que representam as folhas da estrutura hierárquica são unidos, também sem restrições temporais, ao nó de fim da aplicação.

Durante a apresentação, os nós e as arestas do grafo vão sendo marcados. O nó de início da apresentação é o primeiro a ser marcado. Ao ser marcado, as restrições temporais associadas as arestas de saída de um nó geram atrasos que devem ser individualmente respeitados para cada aresta. Quando o tempo relativo a um desses atrasos é atingido, sua aresta é marcada. Quando todas as arestas de entrada de um nó são marcadas, o próprio nó é marcado. A aplicação termina quando o nó de fim da apresentação é marcado.

No grafo de dependências temporais todos os nós correspondentes as folhas da estrutura hierárquica devem ser alcançados, isto é, não existem nós que representam possibilidades que podem ser ou não executadas dependendo, por exemplo, de ações interativas ou adaptativas. Essa estrutura oferece facilidades temporais como a representação do final da apresentação e a indicação de referências circulares (vanRossum, 1993). No entanto, não é possível utilizar diretamente esse grafo no controle de aplicações adaptativas e interativas como, por exemplo, aplicações que podem ser especificadas através de SMIL 3.0 (W3C, 2008b).

Na linguagem SMIL, a estrutura lógica de uma aplicação define a sua estrutura de apresentação. Elementos dessa linguagem representam composições com semântica temporal embutida (paralela, sequencial ou exclusiva) que podem agrupar elementos representando as mídias ou elementos representando outras

composições, recursivamente. Além dessas composições, a linguagem define uma composição voltada à adaptação do conteúdo (*switch*), onde apenas um dos seus elementos constituintes será apresentado (W3C, 2008b). O comportamento temporal das aplicações SMIL também pode ser definido de forma relativa à ocorrência de eventos, sendo possível definir relacionamentos causais entre elementos, de forma complementar a semântica embutida pelas composições. Os eventos associados aos elementos podem ser executados, inclusive, a partir de ações interativas (W3C, 2008b).

Para controlar a apresentação de aplicações SMIL, o exibidor Ambulant (Bulterman, 2004; Cesar, 2006) propõe a construção de uma árvore cujos nós representam todos os elementos da aplicação. As arestas dessa árvore preservam os relacionamentos temporais entre os nós que representam as composições e seus nós filhos, que podem representar as mídias ou outras composições, recursivamente. Outras arestas, representando os relacionamentos existentes entre os eventos da aplicação, também podem existir e, nesse caso, a árvore passa a ser definida como um grafo temporal.

No grafo temporal proposto pelo Ambulant, cada nó possui uma máquina de estados que controla o estado do evento de apresentação da mídia ou da composição associada ao nó. No início da apresentação, o nó correspondente à raiz da árvore tem sua máquina de estados colocada no estado ocorrendo. Quando um nó tem seu estado alterado, essa modificação é notificada aos demais nós relacionados através das arestas. Caso uma aresta tenha alguma condição associada, essa condição deve ser verdadeira para que a notificação aconteça. Uma condição pode ser um atraso temporal, regras em uma adaptação ou ainda um evento interativo.

A estrutura proposta pelo Ambulant é suficiente e eficiente para controlar a apresentação, porém, ela possui características mais próximas da especificação do sincronismo baseada em relacionamentos entre eventos do que da sincronização baseada em grafos. Para calcular os momentos absolutos no tempo relativos à execução dos eventos seria necessário realizar uma simulação da apresentação utilizando essa estrutura, o que não seria viável em tempo de execução da aplicação.

Uma estrutura mais próxima da sincronização baseada em grafos temporais é proposta no sistema Firefly (Buchanan, 1992; Buchanan, 1993; Buchanan,

2005) onde, na verdade, um conjunto de grafos é utilizado para representar os relacionamentos existentes em uma aplicação. A sequência de eventos previsíveis, a partir do evento de apresentação da mídia que inicia a aplicação, forma um grafo nesse sistema denominado principal. Para cada evento imprevisível, cujo tempo de execução não pode ser calculado antes do início da apresentação, um grafo auxiliar é construído. Obviamente, todos os eventos representados em um grafo auxiliar devem ser previsíveis em relação aos demais eventos presentes nesse mesmo grafo.

Durante a apresentação, cada grafo auxiliar pode ser adicionado ao grafo principal tão logo o tempo de execução do evento inicialmente imprevisível que originou esse grafo auxiliar possa ser calculado. Dessa forma, quando o último evento imprevisível da aplicação é conhecido, o grafo temporal é completamente obtido. Esse grafo obtido representa os momentos temporais associados à execução dos eventos em uma aplicação e não à semântica dos relacionamentos entre os eventos. Como exemplo, em aplicações onde eventos em um grafo auxiliar estão relacionados a eventos definidos em outros grafos auxiliares ou ao grafo principal, o grafo obtido pode ser formado por vários eventos com a mesma semântica, porém dispostos em diferentes momentos temporais.

Apêndice B
Comandos de Edição

Comandos	Descrição
openBase (baseId, location)	Abre uma base privada existente localizada pelo parâmetro *location*. Se a base privada não existir ou se o parâmetro location não for informado, uma nova base é criada com o identificador *baseId*. O parâmetro *location* deve obrigatoriamente especificar o dispositivo e o caminho onde está a base a ser aberta
activateBase (baseId)	Ativa uma base privada aberta
deactivateBase (baseId)	Desativa uma base privada aberta
saveBase (baseId, location)	Salva todo o conteúdo da base privada em um dispositivo de armazenamento persistente (se disponível). O parâmetro *location* deve obrigatoriamente especificar o dispositivo e caminho para salvar a base
closeBase (baseId)	Fecha a base privada e descarta todo o seu conteúdo
addDocument (baseId, {uri, id}+)	Adiciona uma aplicação NCL a uma base privada. Os arquivos da aplicação NCL podem ser: a) enviados sem solicitação pela rede de difusão de dados; nesse caso, o par {uri, id} é usado para relacionar um conjunto de caminhos de arquivos especificados na aplicação NCL com suas respectivas localizações no sistema de transporte NOTA: Os conjuntos de pares de referência devem obrigatoriamente ser suficientes para que o middleware possa mapear qualquer referência a arquivos presentes na especificação da aplicação NCL na sua localização concreta na memória do dispositivo receptor. b) recebidos pelo canal de interatividade sob demanda, ou residentes no receptor; para esses arquivos, nenhum par {uri, id} necessita ser enviado, exceto o par {uri, “null”} associado à aplicação NCL, que deverá ser adicionado na base baseId, se a aplicação NCL não for recebida sem solicitação (*pushed file*)
removeDocument (baseId, documentId)	Remove uma aplicação NCL de uma base privada

saveDocument (baseId, documented, location)	Salva uma aplicação NCL em um dispositivo de armazenamento persistente (se disponível). O parâmetro *location* deve especificar o dispositivo e o caminho no dispositivo onde a aplicação será salva. Se a aplicação NCL estiver sendo exibida, ele deve primeiro ser parada (todos os eventos no estado *occurring* devem ser parados)
startDocument (baseId, documentId, interfaceId, offset, refDocumentId, refNodeId) NOTA: O parâmetro offset especifica um valor de tempo.	Inicia a reprodução de uma aplicação NCL em uma base privada, iniciando a apresentação a partir de uma interface específica da aplicação. A referência do tempo transportada no campo *eventNPT* estabelece o ponto de início da aplicação, com respeito à base de tempo NPT do conteúdo *refNodeId* da aplicação *refDocumentId* sendo recebida. Três casos podem ocorrer: 1) Se *eventNPT* for maior ou igual ao valor de NPT da base temporal do conteúdo *refNodeId* sendo recebido, espera-se até que NPT atinja o valor dado em *eventNPT* e começa a exibição da aplicação do seu ponto inícial no tempo+offset; 2) Se *eventNPT* for menor que o valor de NPT da base temporal do conteúdo *refNodeId* sendo recebido, o início da exibição da aplicação é imediato e deslocado no tempo de seu ponto inicial do valor "*offset*+(NPT–*eventNPT*) segundos"; NOTA: Somente nesse caso, o parâmetro offset pode receber um valor negativo, mas *offset*+(NPT–*eventNPT*) segundos deve obrigatoriamente ser um valor positivo 3) Se *eventNPT* for igual a 0, a exibição da aplicação é imediata e a partir de seu ponto inicial no tempo+offset
stopDocument (baseId, documentId)	Pára a apresentação de uma aplicação NCL em uma base privada. Todos os eventos da aplicação que estão em andamento devem obrigatoriamente ser parados
pauseDocument (baseId, documentId)	Pausa a apresentação de uma aplicação NCL em uma base privada. Todos os eventos da aplicação que estão em andamento devem obrigatoriamente ser pausados
resumeDocument (baseId, documentId)	Retoma a apresentação de uma aplicação NCL em uma base privada. Todos os eventos da aplicação que foram previamente pausados pelo o comando de edição *pauseDocument* devem obrigatoriamente ser retomados.
addRegion (baseId, documentId, regionBaseId, regionId, xmlRegion)	Adiciona um elemento <region> como filho de outro <region> no <regionBase>, ou como filho do <regionBase> (*regionId* = "null") de uma aplicação NCL em uma base privada.

removeRegion (baseId, documentId, regionId)	Remove um elemento <region> de um <regionBase> de uma aplicação NCL em uma base privada.
addRegionBase (baseId, documentId, xmlRegionBase)	Adiciona um elemento <regionBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do regionBase for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlRegionBase* é apenas uma referência para esse conteúdo.
removeRegionBase (baseId, documentId, regionBaseId)	Remove um elemento <regionBase> do elemento <head> de uma aplicação NCL em uma base privada.
addRule (baseId, documentId, xmlRule	Adiciona um elemento <rule> ao <ruleBase> de uma aplicação NCL em uma base privada.
removeRule (baseId, documentId, ruleId)	Remove um elemento <rule> do <ruleBase> de uma aplicação NCL em uma base privada.
addRuleBase (baseId, documentId, xmlRuleBase)	Adiciona um elemento <ruleBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do ruleBase for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlRuleBase* é apenas uma referência para esse conteúdo.
removeRuleBase (baseId, documentId, ruleBaseId)	Remove um elemento <ruleBase> do elemento <head> de uma aplicação NCL em uma base privada.
addConnector (baseId, documentId, xmlConnector)	Adiciona um elemento <connector> ao <connectorBase> de um aplicação NCL em uma base privada.
removeConnector (baseId, documentId, connectorId)	Remove um elemento <connector> do <connectorBase> de uma aplicação NCL em uma base privada.
addConnectorBase (baseId, documentId, xmlConnectorBase)	Adiciona um elemento <connectorBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do <connectorBase> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlConnectorBase* é apenas uma referência para esse conteúdo.
removeConnectorBase (baseId, documentId, connectorBaseId)	Remove um elemento <connectorBase> do elemento <head> de uma aplicação NCL em uma base privada.
addDescriptor (baseId, documentId, xmlDescriptor)	Adiciona um elemento <descriptor> ao <descriptorBase> de uma aplicação NCL em uma base privada.
removeDescriptor (baseId, documentId, descriptorId)	Remove um elemento <descriptor> do <descriptorBase> de uma aplicação NCL em uma base privada.

addDescriptorSwitch (baseId, documentId, xmlDescriptorSwitch)	Adiciona um elemento <descriptorSwitch> ao <descriptorBase> de uma aplicação NCL em uma base privada. Se a especificação XML do <descriptorSwitch> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlDescriptorSwitch* é apenas uma referência para esse conteúdo.
removeDescriptorSwitch (baseId, documentId, descriptorSwitchId)	Remove um elemento <descriptorSwitch> do <descriptorBase> de uma aplicação NCL em uma base privada.
addDescriptorBase (baseId, documentId, xmlDescriptorBase)	Adiciona um elemento <descriptorBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do <descriptorBase> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlDescriptorBase* é apenas uma referência para esse conteúdo.
removeDescriptorBase (baseId, documentId, descriptorBaseId)	Remove um elemento <descriptorBase> do elemento <head> de uma aplicação NCL em uma base privada.
addTransition (baseId, documentId, xmlTransition)	Adiciona um elemento <transition> ao <transitionBase> de uma aplicação NCL em uma base privada.
removeTransition (baseId, documentId, transitionId)	Remove um elemento <transition> do <transitionBase> de uma aplicação NCL em uma base privada.
addTransitionBase (baseId, documentId, xmlTransitionBase)	Adiciona um elemento <transitionBase> ao elemento <head> de uma aplicação NCL em uma base privada. Se a especificação XML do <transitionBase> for enviada em um sistema de transporte como um sistema de arquivo, o parâmetro *xmlTransitionBase* é apenas uma referência para esse conteúdo.
removeTransitionBase (baseId, documentId, transitionBaseId)	Remove um elemento <transitionBase> do elemento <head> de uma aplicação NCL em uma base privada.
addImportBase (baseId, documentId, docBaseId, xmlImportBase)	Adiciona um elemento <importBase> à base (elemento <regionBase>, <descriptorBase>, <ruleBase>, <transitionBase>, ou <connectorBase>) de uma aplicação NCL em uma base privada.
removeImportBase (baseId, documentId, docBaseId, documentURI)	Remove um elemento <importBase>, cujo atributo documentURI é identificado pelo parâmetro documentURI, a partir da base (elemento <regionBase>, <descriptorBase>, <ruleBase>, <transitionBase>, or <connectorBase>) de uma aplicação NCL em uma base privada.

addImportedDocumentBase (baseId, documentId, xmlImportedDocumentBase)	Adiciona um elemento <importedDocumentBase> ao elemento <head> de uma aplicação NCL em uma base privada.
removeImportedDocumentBase (baseId, documentId, importedDocumentBaseId)	Remove um elemento <importedDocumentBase> do elemento <head> de uma aplicação NCL em uma base privada.
addImportNCL (baseId, documentId, xmlImportNCL)	Adiciona um elemento <importNCL> ao elemento <importedDocumentBase> de uma aplicação NCL em uma base privada.
removeImportNCL (baseId, documentId, documentURI)	Remove um elemento <importNCL>, cujo atributo documentURI é identificado pelo parâmetro documentURI, a partir do <importedDocumentBase> de uma aplicação NCL em uma base privada.
addNode (baseId, documentId, compositeId, {uri, id}+)	Adiciona um nó (elemento <media>, <context> ou <switch>) a um nó *de composição* (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. A especificação XML do nó e seu conteúdo de mídia podem: a) ser enviados sem solicitação pela rede de difusão de dados; nesse caso, o par {uri, id} é usado para relacionar um conjunto de caminhos de arquivos, definidos no XML da especificação do nó, com suas respectivas localizações no sistema de transporte; NOTA: Os conjuntos de pares de referência devem obrigatoriamente ser suficientes para que o *middleware* possa mapear qualquer referência a arquivos, presentes na especificação do nó, na sua localização concreta na memória do dispositivo receptor b) recebidos pelo canal de interatividade sob demanda, ou residentes no receptor; para esses arquivos, nenhum par {uri, id} necessita ser enviado, exceto o par {uri, “null”} associado à especificação XML do nó NCL que deverá ser adicionado em compositeId, caso o XML não seja recebido sem solicitação (*pushed file*).
removeNode(baseId, documentId, compositeId, nodeId)	Remove um nó (elemento <media>, <context> ou <switch>) de um nó de composição (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada.
addInterface (baseId, documentId, nodeId, xmlInterface)	Adiciona uma interface (<port>, <area>, <property> ou <switchPort>) a um nó (elemento <media>, <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada.

removeInterface (baseId, documentId, nodeId, interfaceId)	Remove uma interface (<port>, <area>, <property> ou <switchPort>) de um nó (elemento <media>, <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada. A *interfaceId* deve obrigatoriamente identificar um atributo name de um elemento <property> ou um atributo id de um elemento <port>, <area> ou <switchPort>.
addLink (baseId, documentId, compositeId, xmlLink)	Adiciona um elemento <link> a um nó de composição (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada.
removeLink (baseId, documentId, compositeId, linkId)	Remove um elemento <link> de um nó de composição (elemento <body>, <context> ou <switch>) de uma aplicação NCL em uma base privada.
setPropertyValue(baseId, documentId, nodeId, propertyId, value)	Atribui o valor a uma propriedade. A *propertyId* deve obrigatoriamente identificar um atributo *name* de um elemento <property> ou um atributo *id* de elemento <switchPort>. O <property> ou <switchPort> deve obrigatoriamente pertencer a um nó (elemento <body>, <context>, <switch> ou <media>) de uma aplicação NCL em uma base privada identificada pelos parâmetros.

Tabela B.1 – Comandos de edição.

Identificadores usados nos comandos	**Descrição**
baseId	Identificador de uma base privada. Usualmente, em um sistema de TV digital, uma base privada é associada a um canal de TV
documentId	Atributo *id* de um <ncl> de uma aplicação NCL
refDocumentId	Atributo *id* de um <ncl> de uma aplicação NCL
refNodeId	Atributo *id* de um <media> de uma aplicação NCL
regionId	Atributo *id* de um <region> de uma aplicação NCL
ruleId	Atributo *id* de um <rule> de uma aplicação NCL
connectorId	Atributo *id* de um <connector> de uma aplicação NCL
descriptorId	Atributo *id* de um <descriptor> de uma aplicação NCL
descriptorSwitchId	Atributo *id* de um <descriptorSwitch> de uma aplicação NCL
transitionId	Atributo *id* de um <transition> de uma aplicação NCL

regionBaseId	Atributo *id* de um <regionBase> de uma aplicação NCL
ruleBaseId	Atributo *id* de um <ruleBase> de uma aplicação NCL
connectorBaseId	Atributo *id* de um <connectorBase> de uma aplicação NCL
descritptorBaseId	Atributo *id* de um <descriptorBase> de uma aplicação NCL
transitionBaseId	Atributo *id* de um elemento <transitionBase> de uma aplicação NCL
docBaseId	Atributo *id* de um <regionBase>, <ruleBase>, <connectorBase>, <descriptorBase>, ou <transitionBase> de uma aplicação NCL
documentURI	Atributo documentURI de um <importBase> ou um elemento <importNCL> de uma aplicação NCL
importedDocumentBaseId	Atributo *id* de um elemento <importedDocumentBase> de uma aplicação NCL
compositeId	Atributo *id* de um <body>, <context> ou <switch> de uma aplicação NCL
nodeId	Atributo *id* de um <body>, <context>, <switch> ou <media> de uma aplicação NCL
interfaceId	Atributo *id* de um <port>, <area>, <property> ou <switchPort> de uma aplicação NCL
linkId	Atributo *id* de um <link> de uma aplicação NCL
propertyId	Atributo *id* de um <property>, ou <switchPort> de uma aplicação NCL

Tabela B.2 – Identificadores utilizados nos comandos de edição.