8
Referências bibliográficas

AKSIT, M. **Aspect oriented software development**. ISBN 0321219767. 2004.

ALMESBERGER, Werner. **Linux network traffic control – Implementation overview**. Implementation details, 1999. Disponível em: <http://snafu.freedom.org/linux2.2/docs/tcio-current.ps>. Acesso em: 24 ago. 2006.

ALMESBERGER, Werner. **Linux Traffic Control - Next Generation**. Linux-Kongress 2002. Disponível em: <http://tcng.sourceforge.net/doc/tcng-overview.pdf>. Acesso em: 24 ago. 2006.

APACHE.ORG. **The Apache Xerces2 Parser 2.9.1**. Apache open source project. 2007. Disponível em: < http://xerces.apache.org/xerces2-j/>. Acesso em: 14 jan. 2008.

BARRETO, L.; MULLER, G. **Bossa: a Language-based Approach for the Design of Real Time Schedulers**. Proceedings of RTS'2002, p.19-31, 2002. Disponível em: <http://www.emn.fr/x-info/bossa/bossa_rts.ps>. Acesso em: 24 ago. 2006.

BARRIA, Marta. **Algoritmos para QoS em Redes de Computadores**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2001. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2001_03_barria.pdf>. Acesso em: 24 ago. 2006.

BAVIER, A. et al. **Operating System Support for Planetary-Scale Network Services**. Proceedings of Networked Systems Design and Implementation 2004 (NSDI'04), 2004. Disponível em: <http://www.planet-lab.org/php/nsdi.pdf>. Acesso em: 24 ago. 2006.

BERSHAD, B. et al. **Extensibility safety and performance in the SPIN operating system**. Proceedings of the fifteenth ACM symposium on Operating systems principles, 1995. Disponível em: <http://portal.acm.org/citation.cfm?id=224077>. Acesso em: 24 ago. 2006.

BLAIR, G. et al. **The Design and Implementation of Open ORB version 2**. IEEE Distributed Systems Online, v.2, n.6, 2001. Disponível em: <http://csdl2.computer.org/comp/mags/ds/2001/06/o6001.pdf>. Acesso em 24 ago. 2006.

BLAKE, S. et al. **An architecture for differentiated services**. IETF Request for Comments (RFC2475), 1998. Disponível em: <http://www.ietf.org/rfc/rfc2475.txt>. Acesso em 24 ago 2006.

BRADEN, R.; CLARK, D.; SHENKER, S. **Integrated services in the internet architecture: an overview**. IETF Request for Comments (RFC1633), 1994. Disponível em: <http://www.ietf.org/rfc/rfc1633.txt>. Acesso em: 24 ago. 2006.

BRAY, T. et al. Extensible Markup Language (XML) 1.0 (Fourth Edition). Recomendação W3C. 2006. Disponível em: < http://www.w3.org/TR/xml/>. Acesso em 14 jan. 2008.

BRUNO, J. et al. **Retrofitting Quality of Service into a Time-Sharing Operating System**. Proceedings of the 1999 USENIX Annual Technical Conference, 1999. Disponível em: <http://www.usenix.org/events/usenix99/ full_papers/bruno/bruno.pdf>.

BUSH, S. Active **Networks and Active Network Management: A Proactive Management Framework**. ISBN 0306465604. 2001.

CAMPBELL, A. et al. (1999) **A Survey of Programmable Networks**. ACM SIGCOMM Computer Communications Review, v.29, n.2, p. 7-23. Disponível em: <http://comet.columbia.edu/genesis/papers/ccr99.pdf>. Acesso em: 24 ago. 2006.

CARDEI, I. **Hierarchical Architecture for Real-Time Adaptive Resource Management**. Proceedings of IFIP/ACM Middleware Conference, 2000. Disponível em: <http://www.cse.fau.edu/~icardei/papers/middleware2k.pdf>. Acesso em 24 ago. 2006.

CHATERJEE, S.; SABATA, B.; BROWN, M. **Adaptive QoS Support for Distributed, Java-Based Applications**. Proceedings of IEEE Int'l Symp. Object-Oriented Real-Time Distributed Computing (ISORC 99), 1999. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/ISORC.1999.776378>. Acesso em: 24 ago. 2006.

CHISNALL, D. **The Definitive Guide to the Xen Hypervisor**. ISBN 013234971X. Prentice Hall. 2007.

COLCHER, Sérgio; GOMES, A. Tadeu; SOARES, L. Fernando. **Um meta modelo para a engenharia de serviços de telecomunicações**. XVIII Simpósio Brasileiro de Redes de Computadores (SBRC'2000), 2000. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/2000_05_colcher.ps.gz>. Acesso em: 24 ago. 2006.

ENGLER, D.; KAASHOEK, F.; O'TOOLE, J. **Exokernel: An operating system architecture for application-level resource management**. Proceedings of the 15th ACM Symposium on Operating System Principles, 1995. Disponível em: <http://www.cpe.virginia.edu/Quals_files/exokernel.pdf>. Acesso em: 24 ago. 2006.

DUBLISH, P. **Service Modeling Language**. Draft Specification, 2006. Disponível em <http://serviceml.org/SML-200607.pdf>. Acesso em 24 ago. 2006.

DRUSCHEL, Peter; BANGA, Gaurav. **Lazy receiver processing (LRP): a network subsystem achitecture for server systems**. Proceedings of 2nd Symposium on Operating System Design and Implementation (OSDI'96), p. 261-275, 1996. Disponível em <http://www.cs.rice.edu/CS/Systems/LRP/osdi96.ps>. Acesso em 24 ago. 2006.

DURAN-LIMON, H.; BLAIR, G. **QoS Management Specification Support for Multimedia Middleware**. J. Systems and Software, v.72, n.1, p.1-23, 2004. Disponível em: <http://www.sciencedirect.com/science/journal/01641212>. Acesso em 24 ago. 2006.

ECLIPSE.ORG. **Eclipse Development Platform - Europa**. Open source project. 2007. Disponível em: <http://www.eclipse.org/downloads/>. Acesso em: 14 jan. 2008.

FLOYD, Sally; JACOBSON, Van. **Link-sharing and Resource Management Models for Packet Networks**. IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, 1995. Disponível em: <http://www.icir.org/floyd/papers/link.pdf>. Acesso em: 24 ago. 2006.

FOSTER, Ian, HESSELMAN, C. e TUECKE, S. **The Anatomy of the Grid: Enabling Scalable Virtual Organizations**. International Journal of High Performance Computing Applications, 15 (3). 200-222, 2001. Disponível em <http://www.globus.org/research/papers/anatomy.pdf>. Acesso em 24 ago. 2006.

FORD, Bryan; SUSARLA, Sai. **CPU inheritance scheduling**. Proceedings of 2nd Symposium on Operating Systems Design and Implementation (OSDI'96), 1996. Disponível em: <http://www.cs.utah.edu/flux/papers/inherit-sched.ps.gz>. Acesso em: 24 ago. 2006.

FreeBSD, **The FreeBSD Project**. Página na Internet. Disponível em: <http://www.freebsd.org/>. 1995-2006. Acesso em: 24 ago. 2006.

GOMES, Antônio T.; COLCHER, Sérgio; SOARES, Luiz F.G. **Towards a Descriptive Approach to Model Adaptable Communication Environments**. Proceedings of IEEE International Conference on Network - ICN2001, 2001. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/2001_06_gomes.pdf>. Acesso em: 24 ago. 2006.

GOTTLIEB, Y.; PETERSON, L. A **Comparative Study of Extensible Routers**. Proceedings of IEEE Open Architectures and Network Programming (OpenArch '02), 2002. Disponível em: <http://www.cs.princeton.edu/nsg/papers/routerstudy_openarch_02/routerstudy-openarch.pdf>. Acesso em: 24 ago. 2006.

GOYAL, Pawan; GUO, Xingang; VIN, Harrick. **A hierarchical CPU scheduler for multimedia operating systems**. Proceedings of 2nd Symposium on Operating System Design and Implementation (OSDI'96), p. 107-122, 1996. Disponível em: <http://www.cs.utexas.edu/users/dmcl/papers/ps/OSDI96.ps>. Acesso em: 24 ago. 2006.

GOYAL, Pawan; VIN, Harrick; CHENG, Haichen. **Start-time fair queuing: a scheduling algorithm for integrated services packet switching networks**. Proceedings of ACM SIGCOMM'96, p. 157-168, 1996. Disponível em: <http://www.cs.utexas.edu/users/dmcl/papers/ps/SIGCOMM96.ps>. Acesso em: 24 ago. 2006.

ISO/IEC 19501. **Unified Modeling Language (UML) Version 1.4.2**. 2005. Disponível em: <http://www.omg.org/cgi-bin/apps/doc?formal/05-04-01.pdf>. Acesso em 14 jan 2008.

ISO/IEC 19757-3. **Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron**. 2006. Disponível em <http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip>. Acesso em 14 jan. 2008.

JIN, J.; NAHRSTEDT, K. **QoS Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy**. IEEE Multimedia, v. 11, ISSN:1070-986X, p. 74-87, 2004. Disponível em: <http://cairo.cs.uiuc.edu/publications/paper-files/ieee_multimedia_jin.pdf>. Acesso em: 24 ago. 2006.

KAVIMANDAN, A.; BALASUBRAMANIAN, K.; SHANKARAN, N.; GOKHALE, A.; SCHMIDT, D.C. **QUICKER: A Model-Driven QoS Mapping Tool for QoS-Enabled Component Middleware**. 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2007. P. 62 – 70, 2007. Disponível em: <http://www.dre.vanderbilt.edu/~amoghk/publications/isorc%202007.pdf>. Acesso em: 14 jan, 2008.

KAY, Michael. **XSL Transformations (XSLT) Version 2.0**. Recomendação W3C. 2007. Disponível em < http://www.w3.org/TR/xslt20/>. Acesso em 14 jan. 2008.

KAY, Michael. **Open Source Saxon XSLT processor Saxon-B 9.0**. Open source project. 2007. Disponível em: <http://saxon.sourceforge.net/>. Acesso em: 14 jan. 2008.

KON, F. et al. **2K: A Distributed Operating System for Dynamic Heterogeneous Environments**. 9th IEEE International Symposium on High Performance Distributed Computing. Pittsburgh, 2000. Disponível em: <http://choices.cs.uiuc.edu/2k/papers/hpdc2000.pdf>. Acesso em: 24 ago. 2006.

KOSMAS, N.; TURNER, K**. Requirements for service creation environments**. Proceedings of 2nd International Workshop on Applied Formal Methods in System Design, p. 133-137, 1997. Disponível em: <http://www.cs.stir.ac.uk/~kjt/research/pdf/req-ser.pdf>. Acesso em 24 ago. 2006.

KOTSOVINOS, Evangelos; ION, Iulia; HARRIS, Tim (2006). **Resource Management for Global Public Computing: Many Policies Are Better Than (N)one**. USENIX WORLDS 2006. Disponível em <http:// research.microsoft.com/~tharris/papers/2006-worlds.pdf>. Acesso em 03 mar. 2007.

KUMAR, Avinesh. **Multiprocessing with the Completely Fair Scheduler - Introducing the CFS for Linux**. IBM DeveloperWorks Article. Documento Eletrônico. 2008. Disponível em: <http://www.ibm.com/developerworks/linux/library/l-cfs/>. Acesso em: 16 mar. 2008.

LAWALL, J. et al. **Bossa Nova: Introducing modularity into the Bossa domain-specific language**. Fourth International Conference on Generative Programming and Component Engineering (GPCE'05), 2005. Disponível em: <http://www.emn.fr/x-info/bossa/gpce05.pdf>. Acesso em 24 ago. 2006.

LAWALL, J.; MULLER, G.; DUCHESNE, H. **Language Design for Implementing Process Scheduling Hierarchies**. ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation - PEPM'04, 2004.

Disponível em: <http://www.emn.fr/x-info/bossa/pepm.pdf>. Acesso em: 24 ago. 2006.

LEE, Chen et. al. **Predictable communication protocol processing in Real-Time Mach**. Proceedings of the Real Time Technology and Applications Symposium, 1996. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=11043&arnumber=509539&count=29&index=23>. Acesso em: 24 ago. 2006.

LESLIE, I. et al. **The design and implementation of an operating system to support distributed multimedia applications**. IEEE Journal on Selected Areas In Communications 14, 7, 1996. Disponível em: <http://www.cl.cam.ac.uk/Research/SRG/netos/old-projects/pegasus/papers/jsac-jun97/paper.html>. Acesso em: 24 ago. 2006.

LIMA, Luciana S.; GOMES, Antônio T.; COLCHER, Sérgio; SOARES, Luiz F.G. **Um Framework para Provisão de QoS em Redes Móveis sem Fio**. XXI Simpósio Brasileiro de Redes de Computadores - SBRC2003, 2003. Disponível em <ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/2003_05_lima_sbrc.pdf>. Acesso em: 24 ago. 2006.

LIU, C.; FLEEMAN, D.; ABER, E.; WELCH, L.; JUEDES, D. **Model-Driven Resource Management for Distributed Real-Time and Embedded Systems**. IEEE Real-Time and Embedded Technology and Applications Symposium, 2004. Disponível em <http://www.cse.wustl.edu/~cdgill/RTAS04/T04MoDES.pdf>. Acesso em 14 jan. 2008.

LIU, C.; LAYLAND, J. **Scheduling algorithms for multiprogramming in a hard-real-time environment**. Journal of the Association for Computing Machinery, v. 20, p. 46-61, 1973. Disponível em: <http://www.acm.org/pubs/articles/journals/jacm/1973-20-1/p46-liu/p46-liu.pdf>. Acesso em 24 ago. 2006.

LOVE, Robert. **Linux Kernel Development**. ISBN 0672327201, Novell Press, 2005.

MILLER, J; MUKERJI, J. MDA **Guide Version 1.0.1**. Overview and guide to OMG's architecture. 2003. Disponível em: < http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf>. Acesso em 14 jan. 2008.

MOSBERGER, D.; PETERSON, L. **Making paths explicit in the scout operating system**. Proceedings of 2nd Symposium on Operating System Design and Implementation (OSDI'96), 1996. Disponível em: <http://portal.acm.org/citation.cfm?id=238771&coll=portal&dl=ACM>. Acesso em: 24 ago. 2006.

MORENO, Marcelo. **Um framework para provisão de QoS em sistemas operacionais**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2002. Disponível em <ftp://ftp.telemidia.puc-rio.br/pub/docs/theses/2002_08_moreno.pdf>. Acesso em 24 ago. 2006.

MORENO, Marcelo, GOMES, Antônio T., COLCHER, Sérgio, SOARES, Luiz F. **Provisão de QoS Adaptável em Sistemas Operacionais: o Subsistema de Rede**. XXI Simpósio Brasileiro de Redes de Computadores (SBRC2003), 2003.

Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/ 2003_05_moreno_sbrc.pdf>. Acesso em: 24 ago. 2006.

MORENO, Marcelo et al. **QoSOS: An adaptable architecture for QoS provisioning in network operating systems**. Journal of the Brazilian Telecommunications Society, Special Issue, v.18, n.2, p.118-131, 2003. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/journalpapers/ 2003_10_moreno.pdf>. Acesso em: 24 ago. 2006.

MORENO, Marcelo; SOARES, Luiz F. **QoSOSLinux: Desenvolvimento de uma extensão Linux com suporte adequado a qualidade de serviço**. I Workshop de Sistemas Operacionais (WSO2004), XXIV Congresso da Sociedade Brasileira de Computação (CSBC2004), 2004. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/ 2004_08_moreno_wso.pdf>. Acesso em 24 ago. 2006.

MORENO, Marcelo; COLCHER, Sergio; SOARES, Luiz F. **Adaptable Resource Management Based on the Virtual Resource Tree Model**. Proceedings of V International Conference on Networking (ICN'06). IEEE Computer Society, 2006. Disponível em: <http://ieeexplore.ieee.org/xpls/ abs_all.jsp?isnumber=34159&arnumber=1628273&count=233&index=35>. Acesso em: 24 ago. 2006.

MORENO, Marcelo; COLCHER, Sérgio; SOARES, Luiz F. **Árvores de Recursos Virtuais: Um modelo de gerenciamento de recursos fim-a-fim com QoS**. XXVII Congresso da SBC. IV Workshop de Sistemas Operacionais (WSO'2007), 2007. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/ conferencepapers/2007_07_moreno_wso.pdf>. Acesso em: 14 jan. 2008.

MOTA, Oscar T.; GOMES, Antônio T.; COLCHER, Sérgio; SOARES, Luiz F.G. **Uma arquitetura adaptável para provisão de QoS na Internet**. XIX Simpósio Brasileiro de Redes de Computadores - SBRC2001, 2001. Disponível em <ftp://ftp.telemidia.puc-rio.br/pub/docs/conferencepapers/2001_05_mota.pdf>. Acesso em: 24 ago. 2006.

NAGAR, S. et al. **Improving Linux resource control using CKRM**. Proceedings of the 2004 Linux Symposium, v.2, 2004. Disponível em: <http://ckrm.sourceforge.net/downloads/ckrm-ols04-paper.pdf>. Acesso em 24 ago. 2006.

NAHRSTEDT, K., CHU, H., NARAYAN, S. **QoS-aware Resource Management for Distributed Multimedia Applications**. Journal on High-Speed Networking, Special Issue on Multimedia Networking, vol. 8, num. 3-4, pp. 227-255, IOS Press, 1998. Disponível em: http://cairo.cs.uiuc.edu/publications/ paper-files/QoSresource.ps>. Acesso em: 24 ago. 2006.

Object Management Group. **Real-time CORBA Architecture**. Final adopted specification, ptc/00-09-02, Object Management Group, 2000. Disponível em: <http://www.omg.org/docs/ptc/00-09-02.pdf>. Acesso em 24 ago. 2006.

Object Management Group. **Object Facility (MOF) Specification**. Final adopted specification, formal/02-04-03, Object Management Group, 2002. Disponível em: < http://www.omg.org/docs/formal/02-04-03.pdf>. Acesso em 24 ago. 2006.

Object Management Group. **Real-time CORBA, v2.0**. Final adopted specification, formal/03-11-01, Object Management Group, 2003. Disponível em: <http://www.omg.org/docs/formal/03-11-01.pdf>. Acesso em 24 ago. 2006.

PAL, P. et al. **Using QDL to Specify QoS-Aware Distributed (QuO) Application Configuration**. Proceedings of 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2K), 2000. Disponível em: <http://www.bbn.com/docs/whitepapers/2000_isorc.pdf>. Acesso em: 24 ago. 2006.

REGEHR, John. **Using Hierarchical Scheduling to Support Soft Real-Time Applications in General-Purpose Operating Systems**. PhD thesis, University of Virginia, 2001. Disponível em: <http://www.cs.utah.edu/~regehr/papers/diss/ regehr-diss-single.pdf>. Acesso em: 24 ago. 2006.

REGEHR, John; STANKOVIC, John. **HLS: A Framework for Composing Soft Real-Time Schedulers**. Proceedings of 22nd IEEE Real-Time Systems Symposium (RTSS2001), p.3-14, 2001. Disponível em: <http://www.cs.utah.edu/ flux/papers/hls-rtss01/RegehrRTSS01.pdf>. Acesso em: 24 ago. 2006.

SALZMAN, P.; BURIAN, M.; POMERANTZ, O. **The Linux Kernel Module Programming Guide v2.6.4**. Página na Internet, 2007. Disponível em: <http://www.tldp.org/LDP/lkmpg/2.6/html/index.html>. Acesso em: 14 jan. 2008.

SELTZER, M.; ENDO, Y.; SMALL, C.; SMITH, A. **Dealing with disaster: Surviving misbehaved kernel extensions**. Proceedings of the 2nd Symposium on Operating Systems Design and Implementation, 1996. Disponível em: <http://portal.acm.org/citation.cfm?id=238779&coll=portal&dl=ACM>. Acesso em 24/08/2006.

SHENOY, P.; VIN, H. Cello: **A Disk Scheduling Framework for Next Generation Operating Systems**", Proceedings of SIGMETRICS'98, ACM, 1998. Disponível em: <http://www.cs.utexas.edu/~vin/pub/pdf/sigmetrics98.pdf>. Acesso em: 24 ago. 2006.

SOARES NETO, Carlos; MORENO, Marcelo F.; GOMES, Antônio T.; SOARES, Luiz F. **Descrição Arquitetural da Provisão de QoS para Suporte a Aplicações Multimídia**. IX Simpósio Brasileiro de Sistemas Multimídia e Web (WebMidia2003), 2003. Disponível em: <ftp://ftp.telemidia.puc-rio.br/pub/docs/ conferencepapers/2003_11_soaresneto_webmidia.pdf>. Acesso em: 24 ago. 2006.

STAHL, Thomas; VÖLTER, Markus. **Model-driven Software Development**. ISBN 0470025700, 2006.

SYNCROSOFT. **<Oxygen/> XML Editor 9.1**. Shareware. Disponível em: < http://www.oxygenxml.com/download.html#dEclipse9.1>. Acesso em: 14 jan. 2008.

SZYPERSKI, C.; MURER, S. **Component Software: Beyond Object-Oriented Programming**. ISBN 0201745720, 2002.

THOMPSON, H.; BEECH, D.; MALONEY, M.; MENDEHLSON, N. **XML Schema Part 1: Structures Second Edition**. Recomendação W3C. 2004. Disponível em < http://www.w3.org/TR/xmlschema-1/>. Acesso em 14 jan. 2008.

WEST, Richard; GLOUDON, Jason. **'QoS Safe' Kernel Extensions for Real-Time Resource Management**. Proceedings of the 14th EuroMicro International Conference on Real-Time Systems, 2002. Disponível em: <http://www.cs.bu.edu/~richwest/papers/ecrts02.pdf>. Acesso em 24 ago. 2006.

XML-DEV Group. **Creating Variable Content Container Elements**. Série XML Schemas: Best Parctices. Documento eletrônico. 2001. Disponível em: <http://www.xfront.com/BestPracticesHomepage.html>. Acesso em 14 jan. 2008.

Anexo I – Schema XML da linguagem Pan

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
      XML Schema for the Pan Language
      Laboratorio TeleMidia - PUC-Rio
      Public URI: http://mdrm.telemidia.puc-rio.br/specs/xml/Pan
-->

<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://mdrm.telemidia.puc-rio.br/specs/xml/Pan"
               xmlns="http://mdrm.telemidia.puc-rio.br/specs/xml/Pan"
  elementFormDefault="qualified" blockDefault="#all">

<!-- ========================= IMPORT ========================= -->
<!-- ImportSetType - Concrete type for document importing -->
   <xs:complexType name="ImportType">
      <xs:attribute name="alias" type="xs:ID" use="required"/>
      <xs:attribute name="src" type="xs:anyURI" use="required"/>
   </xs:complexType>

<!-- ========================= REFERS ========================= -->
<!-- ReferrerType - Concrete type for Referrers of other elements -->
   <xs:complexType name="ReferrerType">
      <xs:attribute name="refer" type="xs:anyURI"/>
   </xs:complexType>

<!-- ReferrerInheritableType - Concrete type for Referrers that can inherit
                              changes made to referred element -->
   <xs:complexType name="ReferrerInheritableType">
      <xs:complexContent>
         <xs:extension base="ReferrerType">
            <xs:attribute name="inherit-changes" type="xs:boolean"
                          default="false"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

<!-- ======================== RESOURCE ======================== -->
<!-- ResourceType - Concrete type for Resource identification -->
   <xs:simpleType name="ResourceType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="cpu"/>
         <xs:enumeration value="network"/>
         <xs:enumeration value="diskspace"/>
         <xs:enumeration value="diskaccess"/>
         <xs:enumeration value="memory"/>
      </xs:restriction>
   </xs:simpleType>

<!-- ===================== GENERIC PARAMS ===================== -->
<!-- ParamType - Abstract Type for a generic parameter -->
   <xs:complexType abstract="true" name="ParamType" block="extension">
      <xs:attribute name="value" use="required" type="xs:anySimpleType"/>
      <xs:attribute name="unit" type="xs:string"/>
   </xs:complexType>

<!-- =================== SERVICE CATEGORIES =================== -->
<!-- param - Abstract element to be replaced by specialized parameters -->
   <xs:element abstract="true" name="servicecategoryparam" type="ParamType"
               block="extension"/>

<!-- ServiceCategoryType - Abstract Type for service categories -->
   <xs:complexType abstract="true" name="ServiceCategoryType"
                   block="extension">
      <xs:choice maxOccurs="unbounded">
         <xs:element ref="servicecategoryparam" maxOccurs="unbounded"/>
      </xs:choice>
   </xs:complexType>

<!-- ===================== GENERIC RULES ====================== -->
<!-- RuleOperatorType - Concrete type for valid rule matching operations -->
   <xs:simpleType name="RuleOperatorType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="eq"/> <!-- equal to                 -->
         <xs:enumeration value="ne"/> <!-- not equal to             -->
         <xs:enumeration value="lt"/> <!-- less than                -->
         <xs:enumeration value="le"/> <!-- less than or equal to    -->
         <xs:enumeration value="gt"/> <!-- greater than             -->
         <xs:enumeration value="ge"/> <!-- greater than or equal to -->
         <xs:enumeration value="rx"/> <!-- regular expression match -->
      </xs:restriction>
   </xs:simpleType>

<!-- RuleType - Abstract type for a single generic rule -->
   <xs:complexType abstract="true" name="RuleType" block="extension">
      <xs:attribute name="op" type="RuleOperatorType" default="eq"/>
      <xs:attribute name="value" type="xs:anySimpleType"/>
   </xs:complexType>

<!-- ================== CLASSIFICATION RULES ================== -->
<!-- classifrule - Abstract element to be replaced by specialized
                  classifrule elements -->
   <xs:element abstract="true" name="classifrule" type="RuleType"
               block="extension"/>

<!-- CompositeClassifRuleType - Concrete type for a compposition of
                               classification rules -->
   <xs:complexType name="CompositeClassifRuleType">
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element name="any" type="CompositeClassifRuleType"/>
         <xs:element name="all" type="CompositeClassifRuleType"/>
         <xs:element ref="classifrule"/>
      </xs:choice>
   </xs:complexType>

<!-- ClassifRulesType - Group of simple and composite classification rules -->
   <xs:complexType name="ClassifRulesType">
      <xs:choice minOccurs="0">
         <xs:element name="any" type="CompositeClassifRuleType"/>
         <xs:element name="all" type="CompositeClassifRuleType"/>
         <xs:element ref="classifrule"/>
      </xs:choice>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>

<!-- ================== ACCESS CONTROL RULES ================== -->
<!-- accessctrlrule - Abstract element to be replaced by specialized
     accessctrlrule elements -->
   <xs:element abstract="true" name="accessctrlrule" type="RuleType"
               block="extension"/>

<!-- CompositeAccessCtrlRuleType - Concrete type for a compposition of
     access control rules -->
   <xs:complexType name="CompositeAccessCtrlRuleType">
      <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element name="any" type="CompositeAccessCtrlRuleType"/>
         <xs:element name="all" type="CompositeAccessCtrlRuleType"/>
         <xs:element ref="accessctrlrule"/>
      </xs:choice>
   </xs:complexType>

<!-- ================= MANAGEMENT STRATEGIES ================== -->
<!-- MgmtStrategyType - Abstract Type for Management strategies -->
   <xs:complexType abstract="true" name="MgmtStrategyType">
      <xs:complexContent>
         <xs:extension base="ReferrerInheritableType">
            <xs:attribute name="id" type="xs:ID"/>
            <xs:attribute name="src" type="xs:anyURI"/>
            <xs:attribute name="type" type="xs:string"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

```
<!-- SchedStrategyType - Concrete Type for scheduling strategies -->
   <xs:complexType name="SchedStrategyType">
      <xs:complexContent>
         <xs:restriction base="MgmtStrategyType">
            <xs:attribute name="type">
               <xs:simpleType>
                  <xs:restriction base="xs:string"/>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>

<!-- AdmStrategyType - Concrete Type for admission strategies -->
   <xs:complexType name="AdmStrategyType">
      <xs:complexContent>
         <xs:restriction base="MgmtStrategyType">
            <xs:attribute name="type">
               <xs:simpleType>
                  <xs:restriction base="xs:string"/>
               </xs:simpleType>
            </xs:attribute>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>

<!-- AccessCtrlRulesType - Group of simple and composite access ctrl rules -->
   <xs:complexType name="AccessCtrlRulesType">
      <xs:choice minOccurs="0">
         <xs:element name="any" type="CompositeAccessCtrlRuleType"/>
         <xs:element name="all" type="CompositeAccessCtrlRuleType"/>
         <xs:element ref="accessctrlrule"/>
      </xs:choice>
      <xs:attribute name="id" type="xs:ID"/>
      <xs:attribute name="inherit-from" type="xs:anyURI"/>
   </xs:complexType>

<!-- ======================== LeafVR'S ======================== -->
<!-- LeafVRAccessCtrlRulesType - Access control rules for Leaf VRs -->
   <xs:complexType name="LeafVRAccessCtrlRulesType">
      <xs:all>
         <xs:element name="tunerules" type="AccessCtrlRulesType"
            minOccurs="0"/>
         <xs:element name="releaserules" type="AccessCtrlRulesType"
            minOccurs="0"/>
         <xs:element name="adaptrules" type="AccessCtrlRulesType"
            minOccurs="0"/>
      </xs:all>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>

<!-- LeafVRMgmtStratsType - Strategies for VR management behavior -->
   <xs:complexType name="LeafVRMgmtStratsType">
      <xs:all>
         <xs:element name="schedstrategy" type="SchedStrategyType"
                     minOccurs="0"/>
         <xs:element name="accessctrlrules" type="LeafVRAccessCtrlRulesType"
                     minOccurs="0"/>
      </xs:all>
   </xs:complexType>

<!-- LeafVRType - Concrete Type for Leaf VR's -->
   <xs:complexType name="LeafVRType">
      <xs:sequence>
         <xs:element name="reservation" type="ServiceCategoryType"
                     block="extension"/>
         <xs:element name="classifrules" type="ClassifRulesType"
                     minOccurs="0"/>
         <xs:element name="mgmtstrategies" type="LeafVRMgmtStratsType"
                     minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attribute name="busy" type="xs:boolean" default="false"/>
   </xs:complexType>

<!-- ======================= InnerVR'S ======================== -->
<!-- ChildVRGroup - Group of child virtual resources that may appear under
                    rootvr or innervr elements -->
   <xs:group name="ChildVRGroup">
```

```
      <xs:choice>
         <xs:element name="innervr" type="InnerVRType"/>
         <xs:element name="leafvr" type="LeafVRType"/>
      </xs:choice>
   </xs:group>

<!-- NonLeafVRAccessCtrlRulesType - Access control rules for Non-Leaf VRs -->
   <xs:complexType name="NonLeafVRAccessCtrlRulesType">
      <xs:all>
         <xs:element name="splitrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="tunerules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="releaserules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="adaptrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
      </xs:all>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>

<!-- NonLeafMgmtStratsType - Strategies for VR management behavior -->
   <xs:complexType name="NonLeafMgmtStratsType">
      <xs:all>
         <xs:element name="schedstrategy" type="SchedStrategyType"
                     minOccurs="0"/>
         <xs:element name="admstrategy" type="AdmStrategyType"
                     minOccurs="0"/>
         <xs:element name="accessctrlrules" type="NonLeafVRAccessCtrlRulesType"
                     minOccurs="0"/>
      </xs:all>
      <xs:attribute name="servicecategory" type="xs:QName" use="required"/>
   </xs:complexType>

<!-- InnerVRType - Concrete Type for Inner VR's -->
   <xs:complexType name="InnerVRType">
      <xs:sequence>
         <xs:element name="reservation" type="ServiceCategoryType"
                     block="extension"/>
         <xs:element name="mgmtstrategies" type="NonLeafMgmtStratsType"/>
         <xs:group ref="ChildVRGroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
   </xs:complexType>

<!-- ====================== ROOT VR'S ====================== -->
<!-- PrimitiveTargetType - Hosts and their resources managed by a
                           Primitive VRT -->
   <xs:complexType name="TargetResourceType">
      <xs:attribute name="type" type="ResourceType" use="required"/>
      <xs:attribute name="host" use="required">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:pattern value="(([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-
5])\.){3}([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
   </xs:complexType>

<!-- TargetVRType - Reference to a LeafVR managed by a Composite VRT -->
   <xs:complexType name="TargetVRType">
      <xs:complexContent>
         <xs:restriction base="ReferrerType">
            <xs:attribute name="refer" use="required"
                          type="xs:anyURI"/>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>

<!-- RootVRType - Abstract Type for Root VR's -->
   <xs:complexType abstract="true" name="RootVRType" block="restriction">
   </xs:complexType>

<!-- RootVRType - Concrete Type for References to Root VR's -->
   <xs:complexType name="ReferVRT">
      <xs:complexContent>
         <xs:extension base="RootVRType">
            <xs:attribute name="refer" type="xs:anyURI" use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

<!-- PrimitiveVRT - Concrete Type for Primtive Root VR's -->
   <xs:complexType name="PrimitiveVRT">
      <xs:complexContent>
         <xs:extension base="RootVRType">
            <xs:sequence>
               <xs:choice>
                  <xs:sequence>
                     <xs:element name="targetresource"
                                 type="TargetResourceType"/>
                     <xs:element name="mgmtstrategies"
                                 type="NonLeafMgmtStratsType"/>
                  </xs:sequence>
                  <xs:element name="targetvr" type="TargetVRType"/>
               </xs:choice>
               <xs:group ref="ChildVRGroup" minOccurs="0"
                         maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

<!-- CompositeVRT - Concrete Type for Composite Root VR's -->
   <xs:complexType name="CompositeVRT">
      <xs:complexContent>
         <xs:extension base="RootVRType">
            <xs:sequence>
               <xs:element name="targetvr" type="TargetVRType"
                           maxOccurs="unbounded"/>
               <xs:element name="mgmtstrategies" type="NonLeafMgmtStratsType"/>
               <xs:group ref="ChildVRGroup" minOccurs="0"
                         maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

<!-- ======================== FORESTS ========================= -->
<!-- ForestAccessCtrlRulesType - Access control rules for Non-Leaf VRs -->
   <xs:complexType name="ForestAccessCtrlRulesType">
      <xs:all>
         <xs:element name="initrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="releaserules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="adaptrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
      </xs:all>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>

<!-- ForestMgmtStratsType - Strategies for VR management behavior -->
   <xs:complexType name="ForestMgmtStratsType">
      <xs:all>
         <xs:element name="admstrategy" type="AdmStrategyType"
            minOccurs="0"/>
         <xs:element name="accessctrlrules" type="ForestAccessCtrlRulesType"
            minOccurs="0"/>
      </xs:all>
   </xs:complexType>

<!-- ForestType - Concrete type for Forests -->
   <xs:complexType name="ForestType">
      <xs:complexContent>
         <xs:extension base="ReferrerType">
            <xs:sequence>
               <xs:element name="mgmtstrategies" type="ForestMgmtStratsType"
                           minOccurs="0"/>
               <xs:choice minOccurs="0" maxOccurs="unbounded">
                  <xs:element name="forest" type="ForestType"/>
                  <xs:element name="rootvr" type="RootVRType"
                     block="restriction substitution"/>
               </xs:choice>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

<!-- ModelType - Concrete type for VRT Models, which are composed of forests  -->
   <xs:complexType name="ModelType">
      <xs:sequence>
         <xs:element name="forest" type="ForestType" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID"/>
   </xs:complexType>

<!-- ====================== MAINTENANCE ======================= -->
<!-- AdaptAccessCtrlRulesType - Access control rules for Non-Leaf VRs -->
   <xs:complexType name="AdaptAccessCtrlRulesType">
      <xs:all>
         <xs:element name="initrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="splitrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="tunerules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="releaserules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
         <xs:element name="adaptrules" type="AccessCtrlRulesType"
                     minOccurs="0"/>
      </xs:all>
   </xs:complexType>

<!-- InitType - Concrete type for Init operations -->
   <xs:complexType name="InitType">
      <xs:choice maxOccurs="unbounded">
         <xs:element name="forest" type="ForestType"/>
         <xs:element name="rootvr" type="RootVRType"
                     block="restriction substitution"/>
      </xs:choice>
      <xs:attribute name="targetforest" type="xs:anyURI" use="required"/>
      <xs:attribute name="id" type="xs:ID" use="required"/>
   </xs:complexType>

<!-- SplitType - Concrete type for Split operations -->
   <xs:complexType name="SplitType">
      <xs:choice>
         <xs:element name="leafvr" type="LeafVRType"/>
         <xs:element name="innervr" type="InnerVRType"/>
      </xs:choice>
      <xs:attribute name="targetvr" type="xs:anyURI" use="required"/>
      <xs:attribute name="id" type="xs:ID" use="required"/>
   </xs:complexType>

<!-- TuneType - Concrete type for Tune operations -->
   <xs:complexType name="TuneType">
      <xs:sequence>
         <xs:element name="reservation" type="ServiceCategoryType"
                     block="extension"/>
      </xs:sequence>
      <xs:attribute name="targetvr" type="xs:anyURI" use="required"/>
      <xs:attribute name="id" type="xs:ID" use="required"/>
   </xs:complexType>

<!-- ReleaseType - Concrete type for Release operations -->
   <xs:complexType name="ReleaseType">
      <xs:attribute name="targetvr" type="xs:anyURI"/>
      <xs:attribute name="targetforest" type="xs:anyURI"/>
      <xs:attribute name="recursive" type="xs:boolean" default="false"/>
      <xs:attribute name="id" type="xs:ID" use="required"/>
   </xs:complexType>

<!-- AdaptType - Concrete type for Adapt operations -->
   <xs:complexType name="AdaptType">
      <xs:sequence>
         <xs:element name="mgmtstrategies">
            <xs:complexType>
               <xs:all>
                  <xs:element name="schedstrategy" type="SchedStrategyType"
                              minOccurs="0"/>
                  <xs:element name="admstrategy" type="AdmStrategyType"
```

```
                              minOccurs="0"/>
                    <xs:element name="accessctrlrules"
type="AdaptAccessCtrlRulesType"
                              minOccurs="0"/>
                 </xs:all>
                 <xs:attribute name="servicecategory" type="xs:QName"/>
              </xs:complexType>
           </xs:element>
        </xs:sequence>
        <xs:attribute name="targetvr" type="xs:anyURI"/>
        <xs:attribute name="targetforest" type="xs:anyURI"/>
        <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>

<!-- TransactionType - Concrete Type for Transactions -->
     <xs:complexType name="TransactionType">
        <xs:choice maxOccurs="unbounded">
           <xs:element name="init" type="InitType"/>
           <xs:element name="split" type="SplitType"/>
           <xs:element name="tune" type="TuneType"/>
           <xs:element name="release" type="ReleaseType"/>
           <xs:element name="adapt" type="AdaptType"/>
        </xs:choice>
        <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>

<!-- OperationResultType - Concrete Type for operation result report -->
     <xs:complexType name="OperationResponseType">
        <xs:sequence>
           <xs:element name="accepted" type="xs:boolean" default="false"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>

<!-- TransactionResponseType - Concrete Type for transaction responses -->
     <xs:complexType name="TransactionResponseType">
        <xs:sequence>
           <xs:choice maxOccurs="unbounded">
              <xs:element name="init-response" type="OperationResponseType"/>
              <xs:element name="split-response" type="OperationResponseType"/>
              <xs:element name="release-response" type="OperationResponseType"/>
              <xs:element name="tune-response" type="OperationResponseType"/>
              <xs:element name="adapt-response" type="OperationResponseType"/>
           </xs:choice>
           <xs:element name="succeeded" type="xs:boolean"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
     </xs:complexType>

<!-- MaintenanceType - Concrete type for maintenance requests -->
     <xs:complexType name="MaintenanceType">
        <xs:sequence>
           <xs:element name="transaction" type="TransactionType"
                       maxOccurs="unbounded"/>
        </xs:sequence>
     </xs:complexType>

<!-- MaintenanceResponseType - Concrete type for maintenance responses -->
     <xs:complexType name="MaintenanceResponseType">
        <xs:sequence>
           <xs:element name="transaction-response" type="TransactionResponseType"
                       maxOccurs="unbounded"/>
        </xs:sequence>
     </xs:complexType>

<!-- ===================== COMPATIBILITY ====================== -->
<!-- SCCompatibilityType - Concrete type for reporting service category
                           compatibility by strategies -->
     <xs:complexType name="SCCompatibilityType">
        <xs:attribute name="servicecategory" type="xs:QName" use="required"/>
     </xs:complexType>

<!-- CompatibilityReportType - Concrete type for compatibility reports for
                               any Pan/MDRM extension -->
     <xs:complexType name="CompatibilityReportType">
        <xs:choice maxOccurs="unbounded">
           <xs:element name="sccompatibility" type="SCCompatibilityType"
                       minOccurs="0"/>
        </xs:choice>
```

```
   </xs:complexType>

<!-- ========================= PANVRT ========================= -->
<!-- panvrt - root element of PanVRT documents -->
   <xs:element name="panvrt">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="title" type="xs:string" minOccurs="0"/>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="import" type="ImportType"
                        minOccurs="0" maxOccurs="unbounded"/>
            <xs:choice>
               <xs:element name="model" type="ModelType"/>
               <xs:element name="maintenance" type="MaintenanceType"/>
               <xs:element name="maintenance-response"
                           type="MaintenanceResponseType"/>
               <xs:element name="compatibility-report"
                           type="CompatibilityReportType"/>
            </xs:choice>
         </xs:sequence>
         <xs:attribute name="id" type="xs:ID"/>
      </xs:complexType>
   </xs:element>

</xs:schema>
```

Anexo II – Regras Schematron da linguagem Pan

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!--
   Schematron rules for the Pan Language
   Laboratorio TeleMidia - PUC-Rio
   Public URI: http://mdrm.telemidia.puc-rio.br/specs/xml/PanSchematron
-->

<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">

   <title>Schematron rules for the Pan Language</title>
   <ns prefix="pan" uri="http://mdrm.telemidia.puc-rio.br/specs/xml/Pan"/>
   <ns prefix="xsi" uri="http://www.w3.org/2001/XMLSchema-instance"/>

   <let name="panns" value="'http://mdrm.telemidia.puc-rio.br/specs/xml/Pan'"/>

<!-- ========================= PANVRT ========================= -->
   <pattern id="docroot-must-be-panvrt">
      <rule context="/*">
         <assert test="node-name(.)=QName($panns,'panvrt')">
            Root of every Pan document should be the 'panvrt' element.
         </assert>
      </rule>
   </pattern>

   <pattern id="required-namespaces">
      <let name="xmlns" value="'http://www.w3.org/2001/XMLSchema-instance'"/>
      <rule context="/pan:panvrt">
         <assert test="//namespace::*[.=$panns] and //namespace::*[.=$xmlns]">
            Namespaces <value-of select="$panns"/> and <value-of select="$xmlns"/>
            should be available!
         </assert>
      </rule>
   </pattern>

<!-- ========================= IMPORT ========================= -->
   <pattern id="import-doc-available">
      <rule context="//pan:import">
         <assert test="doc-available(@src)">
            Imported model could not be resolved or it's not available!
         </assert>
      </rule>
   </pattern>

    <pattern id="import-doc-is-not-self">
      <rule context="//pan:import">
         <let name="imported" value="
            if (doc-available(@src)) then
            doc(@src) else ()"/>
         <report test="$imported=/">
            Imported model cannot be the current document itself!
         </report>
      </rule>
    </pattern>

<!-- ========================== REFER ========================= -->
   <pattern id="refer-can-only-coexist-with-inherit-changes-or-xsitype">
      <rule context="//*[@refer]">
         <report test="@id or @src or @type">
            Attribute "refer" can only coexist with "inherent-changes" or
            "xsi:type"!
         </report>
      </rule>
   </pattern>
   <pattern id="element-with-refer-cannot-have-child-elements">
      <rule context="//*[@refer]">
         <report test="./*">
            Elements with attribute "refer" should not have child elements!
         </report>
      </rule>
   </pattern>

   <pattern id="inherit-changes-needs-a-refer">
      <rule context="//*[@inherit-changes]">
         <assert test="@refer">
            Elements with attribute "inherit-changes" should also have an
            attribute "refer"!
         </assert>
      </rule>
   </pattern>

   <pattern id="refer-correctness">
      <rule context="*[@refer]">
         <let name="aliasref" value="substring-before(@refer,'#')"/>
         <let name="targetref" value="if (contains(@refer,'#')) then
            substring-after(@refer,'#')
            else @refer"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <let name="primitive" value="
            resolve-QName(../@xsi:type,.) = QName($panns,'PrimitiveVRT')"/>
         <let name="nodename" value="
            if (node-name(.)!=QName($panns,'targetvr'))
            then node-name(.)
            else if ($primitive) then QName($panns,'innervr')
            else QName($panns,'leafvr')"/>
         <!-- checks if a local reference is ok -->
         <assert test="
            $aliasref or
            $nodename = node-name(//*[@id=$targetref])">
            Attribute "refer" should point to a <value-of select="$nodename"/>
         </assert>
         <!-- checks if a remote reference is ok -->
         <assert test="
            not ($aliasref) or
            not ($imported) or
            ($imported//*[@id=$targetref] and
             $nodename = node-name($imported//*[@id=$targetref]))">
            Attribute "refer" should point to a <value-of select="$nodename"/>
         </assert>
         <assert test="not($aliasref) or $imported">
            URI in attribute 'refer' could not be resolved.
            Check your import statements.
         </assert>
      </rule>
   </pattern>

   <pattern id="inherit-from-correctness">
      <rule context="*[@inherit-from]">
         <let name="aliasref" value="substring-before(@inherit-from,'#')"/>
         <let name="targetref" value="if (contains(@inherit-from,'#')) then
            substring-after(@inherit-from,'#')
            else @inherit-from"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <let name="nodename" value="node-name(.)"/>
         <!-- checks if a local reference is ok -->
         <assert test="
            $aliasref or
            $nodename = node-name(//*[@id=$targetref])">
            Attribute "inherit-from" should point to a <value-of
select="$nodename"/>
         </assert>
         <!-- checks if a remote reference is ok -->
         <assert test="
            not ($aliasref) or
            not ($imported) or
            ($imported//*[@id=$targetref] and
            $nodename = node-name($imported//*[@id=$targetref]))">
            Attribute "inherit-from" should point to a <value-of
select="$nodename"/>
         </assert>
         <assert test="not($aliasref) or $imported">
            URI in attribute 'inherit-from' could not be resolved.
            Check your import statements.
         </assert>
      </rule>
   </pattern>

<!-- ========================= FORESTS ======================== -->
   <pattern id="forest-cannot-be-a-reference">
      <rule context="//pan:model/pan:forest">
         <report test="@refer">
            A Forest in a model should not have an attribute "refer"
         </report>
      </rule>
   </pattern>

   <pattern id="forests-must-have-id-or-refer">
      <rule context="//pan:forest">
         <report test="not(@id) and not(@refer)">
            Forests should have one of "id" and "refer" attributes!
         </report>
      </rule>
   </pattern>

<!-- ======================= STRATEGIES ======================= -->
   <pattern id="strategies-attributes">
      <rule context="//pan:admstrategy | //pan:schedstrategy">
         <report test="@id and (not(@src and @type))">
            Element <name/> should have "id", "src" and "type" attributes!
         </report>
         <report test="(not(@id) and not(@refer)) or
                       (@id and @refer)">
            Element <name/> should have one of "id" or "refer" attributes!
         </report>
      </rule>
   </pattern>

   <pattern id="strategies-correctness">
      <rule context="//pan:admstrategy | //pan:schedstrategy">
         <let name="aliasref" value="substring-before(@refer,'#')"/>
         <let name="targetref" value="if (contains(@refer,'#')) then
            substring-after(@refer,'#')
            else @refer"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>

         <let name="compreporturi" value="
            if (@src) then concat(@src,'.xml')
            else if (not($aliasref)) then concat(//*[@id=$targetref]/@src,'.xml')
                 else if ($imported) then
                    concat($imported//*[@id=$targetref]/@src,'.xml')
                 else ()"/>
         <let name="compreport" value="
            if (doc-available($compreporturi)) then
               doc($compreporturi)
               else ()"/>

         <let name="sc" value="../@servicecategory"/>
         <let name="context" value="."/>
         <let name="teste" value="resolve-
QName($compreport//pan:sccompatibility/@servicecategory,$compreport/pan:panvrt)"/>
         <assert test="not ($sc) or
                       not($compreport) or
                       $compreport//pan:sccompatibility[
                                    resolve-QName(@servicecategory,.) =
                                    resolve-QName($sc,$context)]">
            This strategy is not compatible with the service category
            specified in 'mgmtstrategies'!<value-of select="$sc"/> <value-of
select="$teste"/>
         </assert>
         <assert test="not($sc) or $compreport">
```

```
            Could not verify service category compatibility for this strategy.
            Attribute 'src' or 'refer' could not be resolved or compatibility
            report does not exist.
         </assert>
      </rule>
   </pattern>

<!-- ========================= ROOTVR ========================= -->
   <pattern id="xsitype-only-for-rootvr-and-reservation">
      <rule context="*[@xsi:type]">
         <assert test="node-name(.) = QName($panns,'rootvr')
                      or node-name(.) = QName($panns,'reservation')">
            Element <name/> cannot have an attribute xsi:type
         </assert>
      </rule>
   </pattern>

   <pattern id="rootvr-composite-or-primitive-or-refer">
      <rule context="//pan:rootvr">
         <assert test="
            @xsi:type and
            (resolve-QName(@xsi:type,.) = QName($panns,'ReferVRT') or
             resolve-QName(@xsi:type,.) = QName($panns,'PrimitiveVRT') or
             resolve-QName(@xsi:type,.) = QName($panns,'CompositeVRT'))">
            Attribute xsi:type should refer to PrimitiveVRT, CompositeVRT
            or ReferVRT types
         </assert>

      </rule>
   </pattern>

<!-- ======================== TARGETVR ======================== -->
   <pattern id="composite-targetvr-single-reference">
      <rule context="//pan:targetvr[resolve-QName(../@xsi:type,.) =
                                   QName($panns,'CompositeVRT')]">
         <let name="refer" value="@refer"/>
         <let name="aliasref" value="substring-before(@refer,'#')"/>
         <let name="targetref" value="if (contains(@refer,'#')) then
                                        substring-after(@refer,'#')
                                     else @refer"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <!-- checks if a local reference is not duplicated -->
         <assert test="
            $aliasref or
            count(//pan:targetvr[@refer=$targetref]) +
            count(//pan:targetvr[@refer=concat('#',$targetref)]) &lt; 2">
            Same leafvr cannot be referred more than once by targetvr's!
         </assert>
         <!-- checks if a remote reference is not duplicated -->
         <assert test="
            not ($aliasref) or
            not ($imported) or
            not ($imported//*[@id=$targetref]/@busy=true() or
                 count(//pan:targetvr[@refer=$refer]) &gt; 1)">
            Same leafvr cannot be referred more than once by targetvr's!
         </assert>
      </rule>
   </pattern>

   <pattern id="primitive-targetvr-correctness">
      <rule context="//pan:targetvr[resolve-QName(../@xsi:type,.) =
                                   QName($panns,'PrimitiveVRT')]">
         <let name="refer" value="@refer"/>
         <let name="aliasref" value="substring-before(@refer,'#')"/>
         <let name="targetref" value="if (contains(@refer,'#')) then
            substring-after(@refer,'#')
            else @refer"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <!-- checks if a local reference has chidren -->
         <assert test="
            $aliasref or
            not ((//*[@id=$targetref]/pan:leafvr or
                  //*[@id=$targetref]/pan:innervr) and
                 (../pan:leafvr or ../pan:innervr))">
            Target VR of this Primtive VRT already has child VRs.
            So, you should specify only the rootvr. You will be able to
            create other child VRs after instantiation.
         </assert>
         <!-- checks if a remote reference has children -->
         <assert test="
            not($aliasref) or
            not (($imported//*[@id=$targetref]/pan:leafvr or
                  $imported//*[@id=$targetref]/pan:innervr) and
                 (../pan:leafvr or ../pan:innervr))">
            Target VR of this Primtive VRT already has child VRs.
            So, you should specify only the rootvr. You will be able to
            create other child VRs after instantiation.
         </assert>
         <!-- Checks if referred innervr is child of a Primitive VRT -->
         <assert test="
            $aliasref or
            //*[@id=$targetref and
                ancestor::pan:rootvr[resolve-QName(@xsi:type,.) =
                QName($panns,'PrimitiveVRT')]]">
            Referred Target VR should be a innervr from a Primitive VRT!
         </assert>
         <!-- Checks if referred innervr is child of a (remote) Primitive VRT -->
         <assert test="
            not($aliasref) or
            $imported//*[@id=$targetref and
            ancestor::pan:rootvr[resolve-QName(@xsi:type,.) =
            QName($panns,'PrimitiveVRT')]]">
            Referred Target VR should be a innervr from a Primitive VRT!
         </assert>
      </rule>
   </pattern>

    <pattern id="leafvr-busy">
      <rule context="//pan:model//pan:leafvr">
         <let name="id" value="@id"/>
         <report test="(not (@busy) or @busy=false()) and
                       //pan:model//pan:targetvr[@refer=$id or
                                                 @refer=concat('#',$id)]">
            This leafvr is being referred by a targetvr in current model.
            Attribute 'busy' should be set to 'true'
         </report>
      </rule>
    </pattern>

<!-- ========================== VR's ========================== -->
   <pattern id="reservation-matches-servicecategory">
      <rule context="//pan:reservation[../../pan:mgmtstrategies]">
         <let name="sc" value="../../pan:mgmtstrategies/@servicecategory"/>
         <assert test="not($sc) or
                       resolve-QName(@xsi:type,.) = resolve-QName($sc,.)">
            Type of 'reservation' must match parent-VR's service category!
         </assert>
      </rule>
      <rule context="//pan:reservation[../../pan:targetvr]">
         <let name="primitive" value="
            resolve-QName(../../@xsi:type,.) = QName($panns,'PrimitiveVRT')"/>
         <let name="refer" value="../../pan:targetvr/@refer"/>
         <let name="aliasref" value="if (contains($refer,'#')) then
            substring-after($refer,'#')
            else $refer"/>
         <let name="targetref" value="substring-
after(../../pan:targetvr/@refer,'#')"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <assert test="$aliasref or not($primitive) or
            resolve-QName(@xsi:type,.) =
            resolve-
QName(//*[@id=$targetref]/pan:mgmtstrategies/@servicecategory,.)">
            Type of 'reservation' should match parent-VR's service category!
         </assert>
         <assert test="not($aliasref) or not($primitive) or
            not($imported) or
            resolve-QName(@xsi:type,.) =
            resolve-
QName($imported//*[@id=$targetref]/pan:mgmtstrategies/@servicecategory,
                          $imported/pan:panvrt)">
            Type of 'reservation' should match parent-VR's service category!
         </assert>
      </rule>
   </pattern>

<!-- ======================= OPERATIONS ======================= -->
   <pattern id="targetvr-attribute-must-be-a-vr">
      <rule context="*[@targetvr]">
         <let name="aliasref" value="substring-before(@targetvr,'#')"/>
         <let name="targetref" value="substring-after(@targetvr,'#')"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <let name="nodename" value="
            if ($imported) then
               node-name($imported//*[@id=$targetref])
            else ()"/>
         <assert test="
            not($imported) or
            QName($panns,'leafvr') = $nodename or
            QName($panns,'innervr') = $nodename or
            QName($panns,'rootvr') = $nodename">
            Attribute "targetvr" should point to a Virtual Resource!
         </assert>
         <assert test="$imported">
            URI in attribute 'targetvr' could not be resolved.
            Check your import statements.
         </assert>

      </rule>
   </pattern>

   <pattern id="targetforest-attribute-must-be-a-forest">
      <rule context="*[@targetforest]">
         <let name="aliasref" value="substring-before(@targetforest,'#')"/>
         <let name="targetref" value="substring-after(@targetforest,'#')"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <let name="nodename" value="
            if ($imported) then
               node-name($imported//*[@id=$targetref])
            else ()"/>
         <assert test="
            not($imported) or
            QName($panns,'forest') = $nodename">
            Attribute "targetforest" should point to a Forest!
         </assert>
         <assert test="$imported">
            URI in attribute 'targetforest' could not be resolved.
            Check your import statements.
         </assert>

      </rule>
   </pattern>

   <pattern id="init-leafvr-busy">
      <rule context="//pan:init//pan:leafvr">
         <let name="id" value="@id"/>
         <report test="(not (@busy) or @busy=false()) and
            ancestor::pan:init//pan:targetvr[@refer=$id or
            @refer=concat('#',$id)]">
            This leafvr is being referred by a targetvr in current operation.
            Attribute 'busy' should be set to 'true'
         </report>
      </rule>
   </pattern>

   <pattern id="adapt-and-release-must-have-targetforest-or-targetvr">
      <rule context="//pan:adapt | //pan:release">
         <report test="(not(@targetforest) and not(@targetvr)) or
                        (@targetforest and @targetvr)">
            Operation '<name/>' should have one of "targetforest" and
```

```
          "targetvr" attributes!
       </report>
    </rule>
 </pattern>

 <pattern id="adapt-correctness">
    <rule context="//pan:adapt/pan:mgmtstrategies/*">
       <let name="aliasref" value="substring-before(../../@targetvr,'#')"/>
       <let name="targetref" value="substring-after(../../@targetvr,'#')"/>
       <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
       <let name="imported" value="
          if (doc-available($importuri)) then
          doc($importuri) else ()"/>
       <let name="nodename" value="
          if ($imported) then
             node-name($imported//*[@id=$targetref])
          else ()"/>
       <report test="
          $imported and ../../@targetvr and
          QName($panns,'admstrategy') = node-name(.) and
          QName($panns,'leafvr') = $nodename">
          Adaptations to Leaf VRs should only involve Scheduling Strategies
          and Access Control Rules
       </report>
       <report test="
          ../../@targetforest and
          QName($panns,'schedstrategy') = node-name(.)">
          Adaptations to Forests should only involve Admission Strategies
          and Access Control Rules
       </report>
    </rule>
 </pattern>

 <pattern id="split-targetvr-must-be-nonleaf">
    <rule context="//pan:split[@targetvr]">
       <let name="aliasref" value="substring-before(@targetvr,'#')"/>
       <let name="targetref" value="substring-after(@targetvr,'#')"/>
       <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
       <let name="imported" value="
          if (doc-available($importuri)) then
          doc($importuri) else ()"/>
       <let name="nodename" value="
          if ($imported) then
             node-name($imported//*[@id=$targetref])
          else ()"/>
       <assert test="
          not($imported) or
          QName($panns,'innervr') = $nodename or
          QName($panns,'rootvr') = $nodename">
          Attribute "targetvr" should point to a remote non-leaf VR!
       </assert>
    </rule>
 </pattern>

 <pattern id="split-matches-servicecategory">
    <rule context="//pan:split/*/pan:reservation">
       <let name="aliasref" value="substring-before(../../@targetvr,'#')"/>
       <let name="targetref" value="substring-after(../../@targetvr,'#')"/>
       <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
       <let name="imported" value="
          if (doc-available($importuri)) then
          doc($importuri) else ()"/>
       <let name="sc" value="
          if ($imported) then
          $imported//*[@id=$targetref]/pan:mgmtstrategies/@servicecategory
          else ()"/>
       <assert test="
          not($imported) or
          resolve-QName(@xsi:type,.) = resolve-QName($sc,$imported/pan:panvrt)">
          Type of 'reservation' should match parent-VR's service category!
       </assert>
    </rule>
 </pattern>

 <pattern id="tune-targetvr-must-be-nonroot">
    <rule context="//pan:tune[@targetvr]">
       <let name="aliasref" value="substring-before(@targetvr,'#')"/>
       <let name="targetref" value="substring-after(@targetvr,'#')"/>
```

```
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <let name="nodename" value="
            if ($imported) then
               node-name($imported//*[@id=$targetref])
            else ()"/>
         <assert test="
            not($imported) or
            QName($panns,'innervr') = $nodename or
            QName($panns,'leafvr') = $nodename">
            Attribute "targetvr" should point to a remote non-root VR!
         </assert>
      </rule>
   </pattern>

   <pattern id="tune-matches-servicecategory">
      <rule context="//pan:tune/pan:reservation">
         <let name="aliasref" value="substring-before(../@targetvr,'#')"/>
         <let name="targetref" value="substring-after(../@targetvr,'#')"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <let name="sc" value="
            if ($imported) then
            $imported//*[@id=$targetref]/../pan:mgmtstrategies/@servicecategory
            else ()"/>
         <assert test="
            not($imported) or
            resolve-QName(@xsi:type,.) = resolve-QName($sc,$imported/pan:panvrt)">
            Type of 'reservation' should match parent-VR's service category!
         </assert>
      </rule>
   </pattern>

   <pattern id="id-uniqueness-after-operation">
      <rule context="//pan:transaction/*//*[@id]">
         <let name="id" value="@id"/>
         <let name="targetvr" value="ancestor::*/@targetvr"/>
         <let name="targetforest" value="ancestor::*/@targetforest"/>
         <let name="aliasref" value="
            if ($targetvr) then substring-before($targetvr,'#')
            else if ($targetforest) then substring-before($targetforest,'#')
            else ()"/>
         <let name="importuri" value="//pan:import[@alias=$aliasref]/@src"/>
         <let name="imported" value="
            if (doc-available($importuri)) then
            doc($importuri) else ()"/>
         <assert test="not ($imported) or not($imported//*[@id=$id])">
            Attribute 'id' should be unique in a model. This 'id' is already
            in use in terget model.
         </assert>
         <assert test="$imported">
            Could not check if attribute 'id' is unique.
         </assert>

      </rule>

   </pattern>

</schema>
```