8
Referências

AGERE SYSTEMS. **Agere network processors**. http://www.agere.com/, 2001. Acesso em: 29 mai. 2004.

AGHA, G. A.; THATI, P. ; ZIAEI, R. **Actors: a Model for Reasoning About Open Distributed Systems**, section 3.1, p. 155–176. Cambridge University Press, New York, EUA, 2001.

ALLEN, R. **A Formal Approach to Software Architecture**. Pittsburgh, Pennsylvania, jan. 1997, 248 p. Tese de doutorado, Carnegie Mellon University, School of Computer Science. Issued as CMU Technical Report CMU-CS-97-144.

ALLEN, R. J.; DOUENCE, R. ; GARLAN, D. Specifying and analysing software architectures. In: INTERNATIONAL CONFERENCE ON FUNDAMENTAL APPROACHES TO SOFTWARE ENGINEERING (FASE98). **Proceedings...** Lisboa, Portugal, mar. 1998.

APPLETON, B. **Sclc and cdiff: Perl scripts for ClearCase**. http://www.cmcrossroads.com/bradapp/clearperl/sclc-cdiff.html, 2003. Acesso em: 9 fev. 2005.

AUJLA, S.; BRUYANT, T. ; SEMMENS, L. Applying formal methods within structured development. **Journal of Selected Areas of Communications**, v. 12, n. 2, p. 258–264, 1994.

AURRECOECHEA, C.; CAMPBELL, A. T. ; HAUW, L. A review of QoS architectures. **ACM Multimedia Systems Journal**, v. 6, n. 3, p. 138–151, nov. 1998.

BASU, A.; HAYDEN, M.; MORRISETT, G. ; VON EICKEN, T. A language-based approach to protocol construction. In: Kamin, S. (Ed.), FIRST ACM SIGPLAN WORKSHOP ON DOMAIN-SPECIFIC LANGUAGES. **Computer Science Reports**. p. 1–15, Urbana, EUA, 1997. University of Illinois.

BATORY, D. S.; GERACI, B. J. Composition validation and subjectivity in GenVoca generators. **IEEE Transactions on Software Engineering**, v. 23, n. 2, p. 67–84, jan. 1997.

BELLISSARD, L.; DE PALMA, N. ; FÉLIOT, D. **The Olan architecture definition language**. Relatório técnico, INRIA Rhône Alpes, Montbonnot Saint-Martain, França, 1998.

BERNDT, H.; GRAUBMANN, P. ; WAKANO, M. Service specification concepts in TINA-C. In: Kugler, H. J.; Mullery, A. ; Niebert, N. (Eds.), SECOND INTERNATIONAL CONFERENCE ON INTELLIGENCE IN SERVICES & NETWORKS (IS&N94). volume 851 de **Lecture Notes in Computer Science**. p. 355–366, Heidelberg, Alemanha, 1994. Springer-Verlag.

BLAIR, L.; BLAIR, G. S. Composition in multiparadigm specification techniques. In: IFIP TC6/WG6.1 THIRD INTERNATIONAL CONFERENCE ON FORMAL METHODS FOR OPEN OBJECT-BASED DISTRIBUTED SYSTEMS (FMOODS). **Proceedings...** p. 401–417, Deventer, Holanda, 1999. Kluwer, B.V.

BLAIR, L.; BLAIR, G. S.; ISSARNY, V.; TUMA, P. ; ZARRAS, A. The role of software architecture in constraining adaptation in component-based middleware platforms. In: INTERNATIONAL CONFERENCE ON MIDDLEWARE. **Proceedings...** 2000.

BLAIR, G. S.; COULSON, G.; ANDERSEN, A.; BLAIR, L.; CLARKE, M.; COSTA, F. M.; DURÁN-LIMÓN, H. A.; FITZPATRICK, T.; JOHNSTON, L.; MOREIRA, R.; PARLAVANTZAS, N. ; SAIKOSKI, K. The design and implementation of OpenORB version 2. **IEEE Distributed Systems Online Journal**, v. 2, n. 6, jun. 2001.

BONACHEA, D.; FISHER, K.; ROGERS, A. ; SMITH, F. Hancock: A language for processing very large-scale data. In: SECOND USENIX CONFERENCE ON DOMAIN-SPECIFIC LANGUAGES. p. 163–176, out. 1999.

BRADEN, R.; ZHANG, L.; BERSON, S.; HERZOG, S. ; JAMIN, S. **RFC2205: Resource ReSerVation Protocol (RSVP) – version 1 functional specification**, 1997.

BRUNETON, E.; COUPAYE, T. ; STEFANI, J. B. Recursive and dynamic software composition with sharing. In: SEVENTH INTERNATIONAL WORKSHOP ON COMPONENT-ORIENTED PROGRAMMING (WCOP02). **Proceedings...** Málaga, Espanha, jun. 2002.

CALDER, M.; KOLBERG, M.; MAGILL, E. H. ; REIFF-MARGANIEC, S. Feature interaction: A critical review and considered forecast. **Computer Networks**, v. 41, n. 1, p. 115–141, jan. 2003.

CAMPBELL, A. T.; DE MEER, H. G.; KOUNAVIS, M. E.; MIKI, K.; VICENTE, J.; VILLELA, D. A. A survey of programmable networks. **ACM SIGCOMM Computer Communications Review**, v. 29, n. 2, p. 7-23, abr. 1999.

CHAN, K.; SELIGSON, J.; DURHAM, D.; GAI, S.; MCCLOGHRIE, K.; HERZOG, S.; REICHMEYER, F.; YAVATKAR, R. ; SMITH, A. **RFC3084: COPS usage for policy provisioning (COPS-PR)**, 2001.

COLCHER, S. **Um metamodelo para aplicações e serviços de comunicação adaptáveis e com qualidade de serviço**. Rio de Janeiro, Brasil, nov. 1999, 125 p. Tese de doutorado, Departamento de Informática - PUC-Rio.

COULSON, G.; BLAIR, G. S.; HUTCHISON, D.; JOOLIA, A.; LEE, K.; UEYAMA, J.; GOMES, A. T. A.; YE, Y. NETKIT: A software component-based approach to programmable networking. **ACM SIGCOMM Computer Communications Review**, v. 33, n. 5, p. 55–66, out. 2003.

COULSON, G.; BLAIR, G. S.; GRACE, P.; JOOLIA, A.; LEE, K. ; UEYAMA, J. OpenCOM v2: a component model for building systems software. In: IASTED SOFTWARE ENGINEERING AND APPLICATIONS. **Proceedings...** Cambridge, Massachusetts, nov. 2004.

DISTRIBUTED MULTIMEDIA RESEARCH GROUP. **DMRG home page**. http://www.comp.lancs.ac.uk/computing/research/mpg/, 2004. Acesso em: 19 fev. 2005.

DA SILVA, S.; FLORISSI, D. ; YEMINI, Y. Composing active services in NetScript. In: DARPA ACTIVE NETWORKS WORKSHOP. **Proceedings...** Tucson, Arizona, mar. 1998.

DASHOFY, E. M.; DER HOEK, A. V. ; TAYLOR, R. N. An infrastructure for the rapid development of XML-based architecture description languages. In: 24TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING. **Proceedings...** p. 266–276, Orlando, Florida, 2002. ACM Press.

DE SOUZA, C. T.; CUNHA, P. R. F. A calculus for reconfigurable component-based systems. In: XXIX CONFERENCIA LATINOAMERICANA DE INFORMATICA. **Proceedings...** set. 2003.

DIETRICH, F. **Modelling Object-Oriented Communcation Services with Temporal Logic**. Lausanne, Suíça, 1999, 183 p. Tese de doutorado, Section de Systèmes de Communication, École Polytechnique Fédérale de Lausanne.

DIETRICH, F.; HUBAUX, J. P. Formal methods for communication services: Meeting the industry expectations. **Computer Networks**, v. 38, n. 1, p. 99–120, jan. 2002.

DURHAM, D.; BOYLE, J.; COHEN, R.; HERZOG, S.; RAJAN, R. ; SASTRY, A. **RFC2748: The COPS (common open policy service) protocol**, 2000.

DURÁN-LÍMON, H. A. **A Resource Management Framework for Reflective Multimedia Middleware**. Lancaster, UK, 2001, 233 p. Tese de doutorado, Lancaster University, Computing Department.

FASSINO, J. P.; STEFANI, J. B.; LAWALL, J. L. ; MULLER, G. Think: A software framework for component-based operating system kernels. In: USENIX ANNUAL TECHNICAL CONFERENCE, GENERAL TRACK. **Proceedings...** p. 73–86. USENIX, 2002.

FONTOURA, M. F. M. C. **A Systematic Approach to Framework Development**. Rio de Janeiro, Brasil, jul. 1999, 165 p. Tese de doutorado, Departamento de Informática - PUC-Rio.

FORMAL SYSTEMS (EUROPE) LIMITED. **The FDR2 model-checker**. http://www.fsel.com/, 2003. Acesso em: 4 fev. 2005.

GOMES, A. T. A. **Um framework para provisão de QoS em ambientes genéricos de processamento e comunicação**. Rio de Janeiro, Brasil, mai. 1999, 162 p. Dissertação de mestrado, Departamento de Informática, PUC-Rio.

GOMES, A. T. A.; COLCHER, S. ; SOARES, L. F. G. Modelling QoS provision on adaptable communication environments. In: INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC). **Proceedings...** p. 1221–1226, Helsinki, Finland, jun. 2001.

____. Towards a descriptive approach to model adaptable communication environments. In: INTERNATIONAL CONFERENCE ON NETWORKING (ICN). Volume 2094 de **Lecture Notes in Computer Science**. p. 867–876, Heidelberg, Alemanha, 2001. Springer-Verlag.

GOMES, A. T. A.; COULSON, G.; BLAIR, G. S.; SOARES, L. F. G. **A component-based approach to the creation and development of network services in the programmable Internet**. MCC 42/03, PUC-Rio, Rio de Janeiro, Brasil, 2003.

GOMES, A. T. A.; SOARES, L. F. G. Uma abordagem de especificação e imposição de restrições de reconfiguração em redes programáveis. In: XXIII SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES. **Anais...** p. 205–218, Fortaleza, Brasil, mai. 2005. Sociedade Brasileira de Computação.

GRASDIJK, M.; DRETELER, J.; BRAUX, H. ; LE-BAIL, J. L. Modelling services in the portfolio from a service provisioning perspective. In: Kugler, H. J.; Mullery, A. ; Niebert, N. (Eds.), SECOND INTERNATIONAL CONFERENCE ON INTELLIGENCE IN SERVICES & NETWORKS (IS&N94). volume 851 de **Lecture Notes in Computer Science**. p. 133–143, Heidelberg, Alemanha, 1994. Springer-Verlag.

HASELTON, E. F. Service creation environments for intelligent networks. **IEEE Communications Magazine**, v. 30, n. 2, p. 78–81, fev. 1992.

HOARE, C. A. R. **Communicating Sequential Processes**. Prentice-Hall International, New Jersey, EUA, 1985.

IERUSALIMSCHY, R.; FIGUEIREDO, L. H.; CELES, W. **Lua 5.0 reference manual**. MCC 14/03, PUC-Rio, Rio de Janeiro, Brazil, 2003.

INTEL CORPORATION. **IXA network processors**. http://www.intel.com/, 2004. Acesso em: 29 mai. 2004.

INTERNATIONAL BUSINESS MACHINE CORPORATION. **PowerNP network processors**. http://www.ibm.com/, 2004. Acesso em: 29 mai. 2004.

INTERNATIONAL ORGANISATION FOR STANDARDISATION. **ISO/IEC 7498: Open systems interconnection – basic reference model**, 1984.

____. **ISO/IEC 8807: Open systems interconnection – LOTOS – a formal description technique based on the temporal ordering of observational behaviour**, 1989.

____. **ISO/IEC 9074: Open systems interconnection – ESTELLE – a formal description technique based on an extended state transition model**, 1989.

____. **ISO/IEC 10746-1: Reference model of open distributed processing, part 1: Overview**, 1995.

____. **ISO/IEC 14977: Information Technology: Syntactic Metalanguage: Extended BNF**, 1996.

____. **Joint Technical Committee 1**. http://www.jtc1.org/, 2001. Acesso em: 25 jun. 2004.

INTERNATIONAL TELECOMMUNICATIONS UNION. **ITU-T Recommendation Z.100: Specification and description language**, 1992.

____. **ITU-T Recommendation I.312/Q.1201: Principles of intelligent network architecture**, 1992.

____. **ITU-T Recommendation Q.700: Introduction to CCITT signalling system no. 7**, 1993.

ISSARNY, V.; BIDAN, C. Aster: a framework for sound customization of distributed runtime systems. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS. **Proceedings...** Hong Kong, Taiwan, mai. 1996.

JAVACC. **Java Compiler Compiler – JavaCC (tm)**. https://javacc.dev.java.net/, 2003. Acesso em: 2 fev. 2005.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. **IEEE Computer**, v. 36, n. 1, p. 41–50, jan. 2003.

KICZALES, G.; DES RIVIÈRES, J. ; BOBROW, D. G. **The Art of the Metaobject Protocol**. MIT Press, Cambridge, EUA, 1991.

KICZALES, G.; LAMPING, J.; MENDHEKAR, A.; MAEDA, C.; LOPES, C. V.; LOINGTIER, J. M. ; IRWIN, J. Aspect-oriented programing. In: ELEVENTH EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING (ECOOP97). volume 1241 de **Lecture Notes in Computer Science**. p. 220–242, Heidelberg, Alemanha, 1997. Springer-Verlag.

KLARLUND, N.; SCHWARTZBACK, M. I. A domain-specific language for regular sets of strings and trees. **IEEE Transactions on Software Engineering**, v. 25, n. 3, p. 378–386, mai. 1999.

KOLBERG, M.; SINNOTT, R. O.; MAGILL, E. H. Experiences modelling and using formal object-oriented telecommunication service frameworks. **Computer Networks**, v. 31, n. 1, 1999.

KONSTANTINOU, A. **Towards autonomic networks**. New York, EUA, 2003, 202 p. Tese de doutorado, Columbia University, Department of Computer Science.

KOSMAS, N.; TURNER, K. J. Requirements for service creation environments. In: Lovrek, I. (Ed.), INTERNATIONAL WORKSHOP ON APPLIED FORMAL METHODS IN SYSTEM DESIGN. **Proceedings...** p. 133–137, Zagreb, Croácia, jun. 1997.

KUGLER, H. J.; MULLERY, A. P.; NIEBERT, N. (Eds.). Towards a pan-european telecommunication service infrastructure, volume 851 de **Lecture Notes in Computer Science**, Heildelberg, Alemanha, 1994. Springer-Verlag.

LABORATÓRIO TELEMÍDIA. **LindaX: An eXtensible description language for adaptable communication systems**. http://www.telemidia.puc-rio.br/products/lindax/, 2004. Acesso em: 2 fev. 2005.

LADD, D. A.; RAMMING, J. C. Two application languages in software production. In: USENIX VERY HIGH LEVEL LANGUAGES SYMPOSIUM. **Proceedings...** p. 169–178, out. 1994.

LOGEAN, X. **Run-time Monitoring and On-line Testing of Middleware-based Communication Services**. Lausanne, Suíça, 1999. Tese de doutorado, Section de Systèmes de Communication, École Polytechnique Fédérale de Lausanne.

LUCKHAM, D.; VERA, J. An event-based architecture definition language. **IEEE Software**, v. 21, n. 9, p. 717–734, set. 1995.

MAES, P. Concepts and experiments in computational reflection. In: ANNUAL CONFERENCE ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES AND APPLICATIONS. **Proceedings...**. p. 147–155, Orlando, Florida, 1987. ACM Press.

MAGEE, J.; KRAMER, J. Dynamic structure in software architectures. In: SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING. **Proceedings...**. p. 3–14, San Francisco, EUA, out. 1996.

MEDVIDOVIC, N.; TAYLOR, R. A classification and comparison framework for software architecture description languages. **IEEE Software**, v. 14, n. 1, p. 70–93, jan. 2000.

MILNER, R. **Calculi for Mobile Processes: the Pi-calculus**. Cambridge University Press, New York, EUA, 1999.

MÖLLER, A.; ÅKERHOLM, M.; FREDRIKSSON, J. ; NOLIN, M. Evaluation of component technologies with respect to industrial requirements. In: 30TH EUROMICRO CONFERENCE. **Proceedings...** p. 56–63, Rennes, França, set. 2004. IEEE Computer Society.

MONROE, R. T.; KOMPANEK, A.; MELTON, R.; GARLAN, D. Architectural styles, design patterns, and objects. **IEEE Software**, v. 14, n. 1, p. 43–52, jan. 1997.

MONROE, R. T. **Capturing software architecture design expertise with Armani**. CMU-CS-98-163, Carnegie Mellon University, Pittsburgh, EUA, 1998.

MORENO, M. F. **Um framework para provisão de QoS em sistemas operacionais**. Rio de Janeiro, Brasil, mai. 2002, 117 p. Dissertação de mestrado, Departamento de Informática, PUC-Rio.

MOREIRA, R. J. S. **FORMAware: Framework of Reflective Components for Managing Architecture Adaptation**. Lancaster, UK, 2003, 213 p. Tese de doutorado, Lancaster University, Computing Department.

MOTA, O. T. J. D. D. L. **Uma arquitetura adaptável para provisão de QoS na Internet**. Rio de Janeiro, Brasil, mai. 2001, 113 p. Dissertação de mestrado, Departamento de Informática, PUC-Rio.

MOY, J. T. **OSPF: Anatomy of an Internet Routing Protocol**. Addison-Wesley Publishing Company, Reading, Massachusetts, 1a. edição, 1998.

MUDHAR, P. A service creation environment for a future intelligent network. In: Kugler, H. J.; Mullery, A. ; Niebert, N. (Eds.), SECOND INTERNATIONAL CONFERENCE ON INTELLIGENCE IN SERVICES & NETWORKS (IS&N94). volume 851 de **Lecture Notes in Computer Science**. p. 333–342, Heidelberg, Alemanha, 1994. Springer-Verlag.

NETWORK PROCESSING FORUM. **History and milestones**. http://www.npforum.org/about/, 2001. Acesso em: 27 set. 2004.

OBJECT MANAGEMENT GROUP. **The Common Object Request Broker: Architecture and specification - revision 2**, 1997.

____. **Unified Modeling Language specification – version 1.5**, 2003.

OBJECTWEB. **The Fractal project**. http://fractal.objectweb.org/, 2004. Acesso em: 11 fev. 2005.

OUSTERHOUT, J. K. **Tcl and the Tk Toolkit**. Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

OUTHRED, G.; POTTER, J. A model for component composition with sharing. In: THIRD INTERNATIONAL WORKSHOP ON COMPONENT-ORIENTED PROGRAMMING (WCOP98). **Proceedings...** p. 29–38, Turku, Finlândia, out. 1998. Turku Centre for Computer Science.

PONTEN, L.; HALLSTRAND, J. ; MARQUES, M. M. Building dedicated service creation environments for reuse-based production. In: Kugler, H. J.; Mullery, A. ; Niebert, N. (Eds.), SECOND INTERNATIONAL CONFERENCE ON INTELLIGENCE IN SERVICES & NETWORKS (IS&N94). volume 851 de **Lecture Notes in Computer Science**. p. 169–178, Heidelberg, Alemanha, 1994. Springer-Verlag.

PREE, W. **Design Patterns for Object-Oriented Software Development**. **ACM Press**. Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

RADISYS CORPORATION. **ENP-2611 data sheet: PCI packet processing engines**. http://www.radisys.com/, 2004. Acesso em: 26 jan. 2005.

RATIONAL SOFTWARE CORPORATION. **Unified modeling language: Notation guide**. http://www.rational.com/uml/resources/documentation, 1997. Acesso em: 19 jan. 2005.

REID, A.; FLATT, M.; STOLLER, L.; LEPREAU, J. ; EIDE, E. Knit: Component composition for systems software. In: FOURTH SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION (OSDI2000). **Proceedings...** p. 347–360, San Diego, California, out. 2000.

ROBLES, T.; HUECAS, G.; QUEMADA, J.; VERDEJO, A.; LLANA-DÍAZ, L. F. **Process calculi: E-LOTOS**, section 2.2, p. 77–104. Cambridge University Press, New York, EUA, 2001.

RODRIGUES, M. A. A. **Um framework para provisão de serviço de multicast em ambientes genéricos de comunicação de dados**. Rio de Janeiro, Brasil, mai. 1999. Dissertação de mestrado, Departamento de Informática, PUC-Rio.

ROSA, N. S.; JUSTO, G. R. R.; CUNHA, P. R. F. An approach for reasoning and refining non-functional requirements. **Journal of the Brazilian Computer Society**, v. 10, n. 1, p. 59–77, 2004.

SCHMIDT, D.; SUDA, T. Transport system architecture services with high-performance communication systems. **Journal of Selected Areas of Communications**, v. 11, n. 4, p. 489–506, 1993.

SCHULZE, B.; NANDKUMAR, R. (Eds.). SECOND INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR GRID COMPUTING (MGC2004), **Proceedings...**, Nova York, EUA, 2004. ACM Press.

SILBERSCHATZ, A.; KORTH, H. F. ; SUDARSHAN, S. **Database System Concepts**. McGraw-Hill Book Company, New York, EUA, 4 edição, 2001.

SINNOTT, R. O.; HOGREFE, D. **Finite State Machine Based: SDL**, section 2.1, p. 55–76. Cambridge University Press, New York, EUA, 2001.

SOARES, L. F. G.; LEMOS, G. ; COLCHER, S. **Redes de Computadores: das LANs, MANs e WANs as Redes ATM**. Editora Campus, Rio de Janeiro, Brasil, 2 edição, 1995.

SOARES-NETO, C. S. **Descrição arquitetural de provisão de QoS em ambientes genéricos de processamento e comunicação**. Rio de Janeiro, Brasil, ago. 2003, 115 p. Dissertação de mestrado, Departamento de Informática - PUC-Rio.

SOARES-NETO, C. S.; MORENO, M. F.; GOMES, A. T. A. ; SOARES, L. F. G. Descrição arquitetural da provisão de QoS para suporte a aplicações multimídia.

In: IX SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E WEB. **Anais...** Salvador, Brasil, nov. 2003. Sociedade Brasileira de Computação

SOARES-NETO, C. S.; RODRIGUES, R. F. ; SOARES, L. F. G. Architectural description of qos provisioning for multimedia application support. In: TENTH INTERNATIONAL MULTIMEDIA MODELLING CONFERENCE. **Proceedings...** p. 161–166, Brisbane, Australia, jan. 2004.

SZYPERSKI, C. **Component Software: Beyond Object-oriented Programming**. Addison-Wesley Publishing Company, Reading, Massachusetts, 2a. edição, 2002.

TAYLOR, R. N.; MEDVIDOVIC, N.; ANDERSON, K. M.; WHITEHEAD, E. J.; ROBBINS, J. E.; NIES, K. A.; OREIZY, P.; DUBROW, D. A component- and message-based architectural style for GUI software. **IEEE Transactions on Software Engineering**, v. 22, n. 6, p. 390–406, jun. 1996.

TENNENHOUSE, D. L.; SMITH, J. M.; SINCOSKIE, W. D.; WETHERALL, D. J.; MINDEN, G. J. A survey of active network research. **IEEE Communications Magazine**, v. 35, n. 1, p. 80–86, jan. 1997.

THE REGENTS OF THE UNIVERSITY OF CALIFORNIA. **ArchStudio 3: a software architecture-based development environment**. http://www.isr.uci.edu/projects/archstudio/, 2004. Acesso em: 9 fev. 2004.

UEYAMA, J.; COULSON, G.; BLAIR, G. S.; SCHMID, S.; GOMES, A. T. A.; JOOLIA, A.; LEE, K. A globally-applied component model for programmable networking. In: Wakamiya, N.; Solarski, M.; Sterbenz, J. P. G. (Eds.), FIFTH INTERNATIONAL WORKING CONFERENCE ON ACTIVE NETWORKS (IWAN2003). volume 2982 de **Lecture Notes in Computer Science**. p. 202–214, Heidelberg, Alemanha, 2003. Springer-Verlag.

VAN DEURSEN, A.; KLINT, P. ; VISSER, J. Domain-specific languages: An annotated bibliography. **ACM SIGPLAN Notices**, v. 35, n. 6, p. 26–36, jun. 2000.

WORLD WIDE WEB CONSORTIUM. **XML schema part 0: Primer**. http://www.w3.org/TR/xmlschema-0, 2001. Acesso em: 19 jan. 2005.

WALL, L.; SCHWARTZ, R. L. **Programming perl**. O'Reilly & Associates, Inc., 1991.

WROCLAWSKI, J. **RFC22210: The use of RSVP with IETF integrated services**, 1997.

ZAREMSKI, A. M.; WING, J. M. Specification matching of software components. **ACM Transactions on Software Engineering and Methodology**, v. 6, n. 4, p. 333–369, out. 1997.

ZNATY, S.; HUBAUX, J. P. Telecommunications services engineering: Definitions, architectures and tools. In: Bosch, J.; Mitchell, S. (Eds.), ECOOP 97 WORKSHOPS. volume 1357 de **Lecture Notes in Computer Science**. p. 3–11, Jyvaskyla, Finland, fev. 1998. Springer.

9
Apêndice A

Neste apêndice são apresentados alguns detalhes de implementação das ferramentas do ambiente LindaStudio e da instanciação do *framework* para gerência de adaptações na plataforma OpenCOM.

9.1.
Implementação do LindaStudio

O ambiente LindaStudio consiste em aproximadamente 35.000 linhas de código Java (desconsiderando linhas delimitadoras e comentários[1]), distribuídas por 29 pacotes e 312 arquivos-fonte. Cerca de 40% desse código é dedicado à integração de LindaStudio com o ambiente ArchStudio 3.

Originalmente, o ambiente ArchStudio 3 oferece uma ADL, chamada xADL (Dashofy et al., 2002), que é definida a partir de um conjunto de esquemas XML derivados do esquema central `xArch`, bem como um conjunto de ferramentas específicas para essa linguagem. Para tornar o ambiente LindaStudio mais "leve"[2], as ferramentas do ambiente ArchStudio específicas para xADL foram removidas.

As telas dos *drivers* das ferramentas `Translator` e `Generator` são apresentadas na Figura 9.1. A Figura 9.2 ilustra uma captura de tela com as outras ferramentas que compõem o ambiente LindaStudio: `FileManager/Invoker`, `ArchEdit`, `CriticGUI` (integrantes do ArchStudio), `StyleEditor` e `ConfEditor` (introduzidas pelo LindaStudio).

Os editores de estilos e configurações (`StyleEditor` e `ConfEditor`) tiveram seus *parsers* da notação sem *tags* introduzida no Capítulo 3 gerados e implementados com a biblioteca JavaCC (2003).

[1]Para obter essa estatística, foi usada a ferramenta `sclc` (Appleton, 2003), com as opções `-delim-ignore` e `-counts ncsl`.

[2]Para sistemas complexos, o estilo C2 pode impor uma sobrecarga considerável de processamento devido ao excesso de mensagens difundidas pelos conectores-barramento sendo recebidas e descartadas por componentes para os quais essas mensagens não são relevantes.

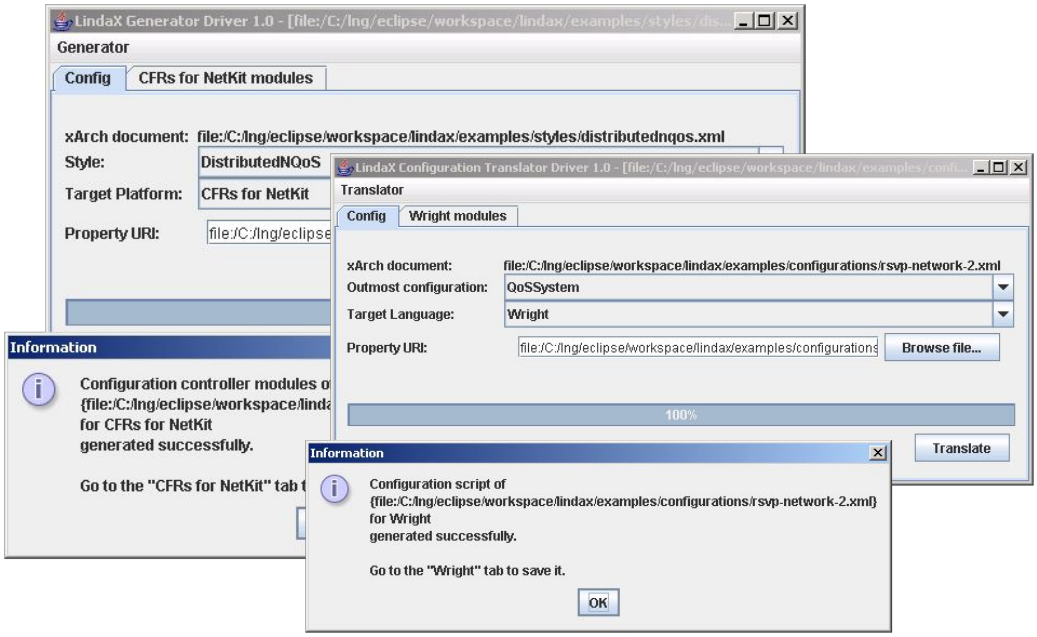

Figura 9.1. *Drivers* das ferramentas `Translator` e `Generator`.

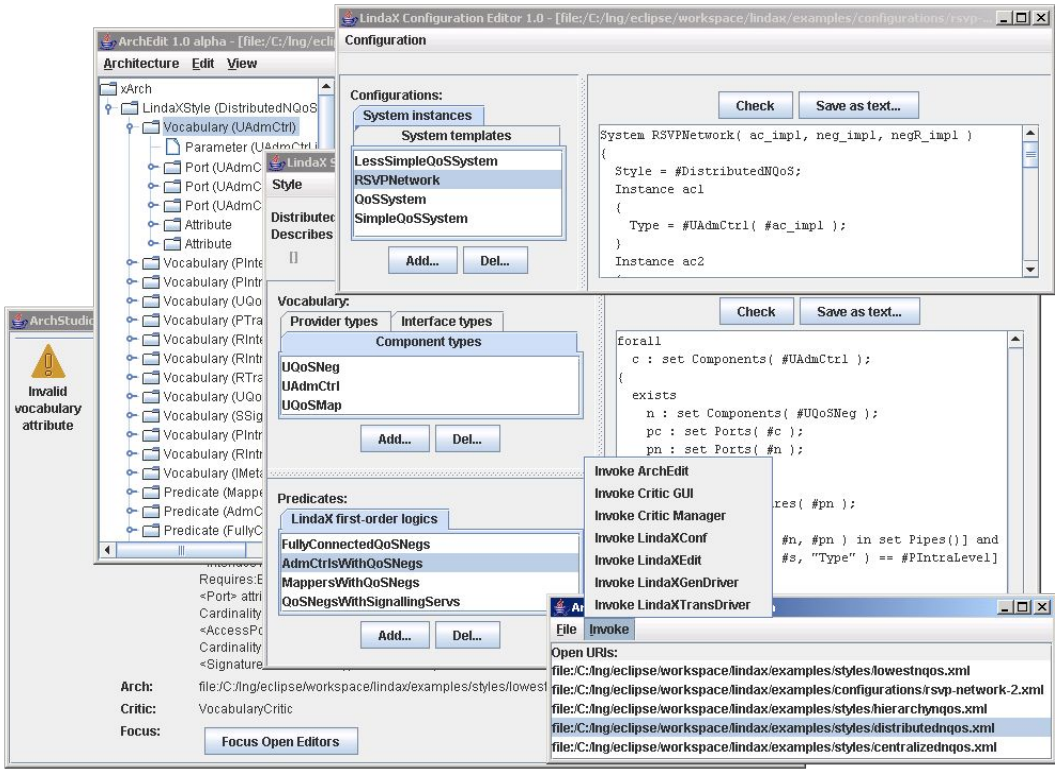

Figura 9.2. Captura de tela do ambiente LindaStudio.

A seguir são apresentadas as interfaces Java das ferramentas `Generator`, `Translator` e `Expander`.

9.1.1. Interface de `Generator`

```
11 public abstract interface IGenerator {
12   /* retorna lista de scripts de verificação */
13   protected abstract Hashtable generate(
14    String url,    // localização do documento LindaX
15    String objId,  // ID do estilo
16    Hashtable symb // lista de valores para propriedades parametrizadas
17   ) throws Exception;
18 }
```

9.1.2. Interface de `Translator`

```
1  public abstract interface ITranslator {
2    /* retorna lista de scripts de configuração */
3    protected abstract Hashtable translate(
4     String url,    // localização do documento LindaX
5     String objId,  // ID do sistema ou template
6     Hashtable symb // lista de valores para propriedades parametrizadas
7    ) throws Exception;
8  }
```

9.1.3. Interface de `Expander`

```
1  public abstract interface IExpander {
2    /* retorna estrutura com informação sobre o conjunto de instâncias
3       inferidas a partir de uma cláusula “Instance” */
4    public abstract InstanceInfo expandInstance(
5     ObjRef instRef // ref. a um elemento XML de um documento LindaX
6                    //  que representa uma instância
7    ) throws Exception;
8
9    /* retorna estrutura com informação sobre o tipo do pipe
10      e sobre quais portas dos componentes estão ligadas a ele,
11      inferidos a partir de uma cláusula “Pipe” */
12   public abstract PipeInfo expandPipe(
13    ObjRef pipeRef // ref. a um elemento XML de um documento LindaX
14                   //  que representa um pipe
15   ) throws Exception;
16
17   /* retorna estrutura com informação sobre o tipo da porta
18      externalizada por um componente ou sistema,
19      inferido a partir de uma cláusula “Mapping” */
20   public abstract PortInfo expandMapping(
21    ObjRef mapRef // ref. a um elemento XML de um documento LindaX
22                  //  que representa um mapeamento
23   ) throws Exception;
24 }
25
26 public class InstanceInfo {
27   public CompInfo[] eInfos;
28   public PipeInfo[] aInfos;
29 }
30
31 public class CompInfo {
32   public String compName;    // nome do componente
33   public ObjRef compTypeRef; // ref. a um elemento XML de um
34                              //  documento LindaX que representa um
35                              //  tipo de componente
36 }
```

```
37
38 public class PipeInfo {
39   public PortInfo[] pInfos;
40   public ObjRef pipeTypeRef; // ref. a um elemento XML de um
41                              //  documento LindaX que representa um
42                              //  tipo de pipe
43 }
44
45 public class PortInfo {
46   public ObjRef[] instRefs; // refs. a elementos XML de um (ou mais)
47                             //  documento(s) LindaX que representam
48                             //  o aninhamento de um conjunto de
49                             //  instâncias (o último elemento da lista
50                             //  é instância de componente primitivo)
51   public String instName;   // nome da instância mais
52                             //  EXTERNA no aninhamento
53   public ObjRef typeRef;    // tipo da instância mais
54                             //  INTERNA no aninhamento
55   public ObjRef[] portRefs; // refs. a elementos XML de um
56                             //  documento LindaX que representam
57                             //  as portas externalizadas pela
58                             //  instância mais INTERNA no aninhamento
59 }
```

9.2. Instanciação do *framework* para gerência de adaptações no OpenCOM

O *framework* consiste em aproximadamente 3.000 linhas de código C/C++ (desconsiderando linhas delimitadoras e comentários[3]), distribuídas por 3 pacotes e 19 arquivos-fonte. A distribuição desse código entre os vários componentes do *framework* é apresentada na Tabela 9.1. Essa tabela discrimina quantas linhas de código são dedicadas à lógica e aos *skeletons* em C++ das interfaces OpenCOM de cada um dos componentes. No caso do GAC, é discriminado também o número de linhas de código da API Lua disponibilizada por esse componente aos *scripts* de verificação.

Tabela 9.1. Número de linhas de código de cada componente do *framework*.

Componente	Linhas de código
GAC *Skeleton* *API Lua*	563 *118* *396* (total = 1077)
TC *Skeleton*	1161 *91* (total = 1252)
CG *Skeleton*	460 *38* (total = 498)

[3]Novamente, foi usada a ferramenta `sclc`, com as opções `-delim-ignore` e `-counts ncsl`, no cômputo dessa estatística.

A seguir são apresentadas as IDLs das interfaces OpenCOM disponibilizadas pelos componentes do *framework*.

9.2.1. Interfaces do GAC

```
1  interface ICFControl {
2    void addConstraint( [const] in  string        script,
3                        [const] in  string        constraintName,
4                                out unsigned long constraintID );
5
6    void removeConstraint( in unsigned long constraintID );
7
8    void enumConstraints(      out unsigned long iCount,
9      [array, size_is(iCount)] out unsigned long constraintIDs );
10
11   void getConstraintByID( in  unsigned long constraintID,
12                           out string        constraintName );
13
14   void getConstraintByName( [const] in  string        constraintName,
15                                     out unsigned long constraintID );
16
17   void removeAllConstraints();
18
19   void replaceConstraintScript( in unsigned long constraintID,
20     [const]                     in string        script );
21 };
22
23 interface ICFValidation {
24   void validate(             out unsigned long uCount,
25     [array, size_is(uCount)] out unsigned long unsatisfConstraints );
26 };
```

9.2.2. Interface do TC

```
1  interface ICFTransaction {
2    void begin(    out unsigned long transactionID );
3    void commit(   in  unsigned long transactionID );
4    void rollback( in  unsigned long transactionID );
5  };
```

9.2.3. Interface do CG

```
1  interface ICFConfiguration {
2    void configure( [const] in string script );
3  };
```

10
Apêndice B

Este apêndice apresenta a especificação completa dos estilos associados à DSL LindaQoS.

10.1.
Estilo `LowestNQoS`

```
4  Style LowestNQoS {
5    InterfaceType IMetaQoS( impl,behv ) {
6      Implementation = #impl;
7      Behaviour = #behv;
8    }
9    InterfaceType IResourceMan( impl,behv ) {
10     Implementation = #impl;
11     Behaviour = #behv;
12   }
13   InterfaceType PInterLevel( impl,behv ) {
14     Implementation = #impl;
15     Behaviour = #behv;
16   }
17
18   ComponentType UAdmCtrl( impl,behv ) {
19     Implementation = #impl;
20     Behaviour = #behv;
21     Cardinality = { 1 ..  };
22
23     Port interLevel {
24       Cardinality = { 1 ..  };
25       Type = #PinterLevel;
26       Direction = “in”;
27     }
28     Port resourceMan {
29       Cardinality = 1;
30       Level = "meta";
31       Type = #IResourceMan;
32       Direction = “out”;
33     }
34     Port metaQoS {
35       Cardinality = { 0 .. 1 };
36       Level = "meta";
37       Type = #IMetaQoS;
38       Direction = “in”;
39     }
40   }
41 }
```

10.2. Estilo `CentralizedNQoS`

```
1  Style CentralizedNQoS {
2    InterfaceType IMetaQoS( impl,behv ) {
3      Implementation = #impl;
4      Behaviour = #behv;
5    }
6    InterfaceType PIntraLevel( impl,behv ) {
7      Implementation = #impl;
8      Behaviour = #behv;
9    }
10   InterfaceType PInterLevel( impl,behv ) {
11     Implementation = #impl;
12     Behaviour = #behv;
13   }
14   InterfaceType PTranslate( impl,behv ) {
15     Implementation = #impl;
16     Behaviour = #behv;
17   }
18
19   ComponentType UQoSNeg( impl,behv ) {
20     Implementation = #impl;
21     Behaviour = #behv;
22     Cardinality = 1;
23
24     Port intraLevel {
25       Cardinality = { 1 ..  };
26       Type = #PIntraLevel;
27       Direction = “in”;
28     }
29     Port interLevel {
30       Cardinality = { 1 ..  };
31       Type = #PInterLevel;
32       Direction = “out”;
33     }
34     Port translate {
35       Cardinality = { 1 ..  };
36       Type = #PTranslate;
37       Direction = “out”;
38     }
39     Port metaQoS {
40       Cardinality = { 0 .. 1 };
41       Level = "meta";
42       Type = #IMetaQoS;
43       Direction = “in”;
44     }
45   }
46
47   ComponentType UAdmCtrl( impl,behv ) {
48     Implementation = #impl;
49     Behaviour = #behv;
50     Cardinality = { 1 ..  };
51
52     Port interLevel {
53       Cardinality = { 1 ..  };
54       Type = #PInterLevel;
55       Direction = “in”;
56     }
57     Port intraLevel {
58       Cardinality = 1;
59       Type = #PIntraLevel;
60       Direction = “out”;
61     }
62     Port metaQoS {
63       Cardinality = { 0 .. 1 };
64       Level = "meta";
65       Type = #IMetaQoS;
66       Direction = “in”;
67     }
68   }
69
```

```
70  ComponentType UQoSMap( impl,behv ) {
71    Implementation = #impl;
72    Behaviour = #behv;
73    Cardinality = { 1 ..  };
74
75    Port translate {
76      Cardinality = 1;
77      Type = #PTranslate;
78      Direction = “in”;
79    }
80  }
81
82  PipeType IntraLevelPipe( impl,behv ) {
83    Implementation = #impl;
84    Behaviour = #behv;
85    Cardinality = { 1 .. };
86
87    AccessPoint in {
88      Cardinality = 1;
89      Type = #PIntraLevel;
90      Direction = “in”;
91    }
92    AccessPoint out {
93      Cardinality = 1;
94      Type = #PIntraLevel;
95      Direction = “out”;
96    }
97  }
98
99  PipeType TranslatePipe( impl,behv ) {
100   Implementation = #impl;
101   Behaviour = #behv;
102   Cardinality = { 1 .. };
103
104   AccessPoint in {
105     Cardinality = 1;
106     Type = #PTranslate;
107     Direction = “in”;
108   }
109   AccessPoint out {
110     Cardinality = 1;
111     Type = #PTranslate;
112     Direction = “out”;
113   }
114 }
115
116 // Todo metacomponente de mapeamento
117 // deve estar ligado a um metacomponente de negociação central.
118 FolPredicate MappersWithQoSNeg {
119   exists n : Components( #UQoSNeg ); {
120     forall c : Components( #UQoSMap ); {
121       exists pn : Ports( #n );
122              pc : Ports( #c ); {
123         exists s : Signatures( #pc ); {
124           [Pipe( #n,#pn,#c,#pc ) in Pipes()] and
125           [PropertyValue( #s,“Type” ) == #PTranslate]
126         }
127       }
128     }
129   }
130 }
131
132 // Todo metacomponente de controle de admissão
133 // deve estar ligado a um metacomponente de negociação central.
134 FolPredicate AdmCtrlsWithQoSNeg {
135   exists n : Components( #UQoSNeg ); {
136     forall c : Components( #UAdmCtrl ); {
137       exists pc : Ports( #c );
138              pn : Ports( #n ); {
139         exists s : Signatures( #pn ); {
140           [Pipe( #c,#pc,#n,#pn ) in Pipes()] and
141           [PropertyValue( #s,“Type”) == #PIntraLevel]
142         }
```

```
143         }
144       }
145     }
146   }
147}
```

10.3. Estilo `DistributedNQoS`

```
1  Style DistributedNQoS {
2    InterfaceType IMetaQoS( impl,behv ) {
3      Implementation = #impl;
4      Behaviour = #behv;
5    }
6    InterfaceType PIntraLevel( impl,behv ) {
7      Implementation = #impl;
8      Behaviour = #behv;
9    }
10   InterfaceType PInterLevel( impl,behv ) {
11     Implementation = #impl;
12     Behaviour = #behv;
13   }
14   InterfaceType PIntraNeg( impl,behv ) {
15     Implementation = #impl;
16     Behaviour = #behv;
17   }
18   InterfaceType PTranslate( impl,behv ) {
19     Implementation = #impl;
20     Behaviour = #behv;
21   }
22
23   ComponentType UQoSNeg( impl,behv ) {
24     Implementation = #impl;
25     Behaviour = #behv;
26     Cardinality = {2 .. };
27
28     Port intraLevel {
29       Cardinality = { 0 ..  };
30       Type = #PIntraLevel;
31       Direction = “in”;
32     }
33     Port interLevel {
34       Cardinality = { 1 ..  };
35       Type = #PInterLevel;
36       Direction = “out”;
37     }
38     Port intraNeg {
39       Cardinality = { 1 ..  };
40       Signature in {
41         Type = #PIntraNeg;
42         Direction = “in”;
43       }
44       Signature out {
45         Type = #PIntraNeg;
46         Direction = “out”;
47       }
48     }
49     Port translate {
50       Cardinality = { 1 ..  };
51       Type = #PTranslate;
52       Direction = “out”;
53     }
54     Port metaQoS {
55       Cardinality = { 0 .. 1 };
56       Level = "meta";
57       Type = #IMetaQoS;
58       Direction = “in”;
59     }
60   }
```

```
61
62   ComponentType UAdmCtrl( impl,behv ) {
63     Implementation = #impl;
64     Behaviour = #behv;
65     Cardinality = { 1 ..  };
66
67     Port interLevel {
68       Cardinality = { 1 ..  };
69       Type = #PInterLevel;
70       Direction = “in”;
71     }
72     Port intraLevel {
73       Cardinality = 1;
74       Type = #PIntraLevel;
75       Direction = “out”;
76     }
77     Port metaQoS {
78       Cardinality = { 0 .. 1 };
79       Level = "meta";
80       Type = #IMetaQoS;
81       Direction = “in”;
82     }
83   }
84
85   ComponentType UQoSMap( impl,behv ) {
86     Implementation = #impl;
87     Behaviour = #behv;
88     Cardinality = { 1 ..  };
89
90     Port translate {
91       Cardinality = 1;
92       Type = #PTranslate;
93       Direction = “in”;
94     }
95   }
96
97   PipeType IntraLevelPipe( impl,behv ) {
98     Implementation = #impl;
99     Behaviour = #behv;
100    Cardinality = { 1 .. };
101
102    AccessPoint in {
103      Cardinality = 1;
104      Type = #PIntraLevel;
105      Direction = “in”;
106    }
107    AccessPoint out {
108      Cardinality = 1;
109      Type = #PIntraLevel;
110      Direction = “out”;
111    }
112  }
113
114  PipeType TranslatePipe( impl,behv ) {
115    Implementation = #impl;
116    Behaviour = #behv;
117    Cardinality = { 1 .. };
118
119    AccessPoint in {
120      Cardinality = 1;
121      Type = #PTranslate;
122      Direction = “in”;
123    }
124    AccessPoint out {
125      Cardinality = 1;
126      Type = #PTranslate;
127      Direction = “out”;
128    }
129  }
130
131  PipeType SignallingPipe( impl,behv ) {
132    Implementation = #impl;
133    Behaviour = #behv;
```

```
134    Cardinality = { 1 .. };
135
136    AccessPoint intraNeg {
137      Cardinality = { 2 ..  };
138       Signature in {
139         Type = #PIntraNeg;
140         Direction = “in”;
141       }
142       Signature out {
143         Type = #RIntraNeg;
144         Direction = “in”;
145       }
146    }
147  }
148
149  // Todo metacomponente de mapeamento
150  // deve estar ligado a um metacomponente de negociação.
151  FolPredicate MappersWithQoSNegs {
152    forall c : Components( #UQoSMap ); {
153      exists n : Components( #UQoSNeg ); {
154        exists pn : Ports( #n );
155               pc : Ports( #c ); {
156          exists s : Signature( #pc ); {
157            [Pipe( #n,#pn,#c,#pc ) in Pipes()] and
158            [PropertyValue( #s,“Type” ) == #PTranslate]
159          }
160        }
161      }
162    }
163  }
164
165  // Todo metacomponente de controle de admissão
166  // deve estar ligado a um metacomponente de negociação.
167  FolPredicate AdmCtrlsWithQoSNegs {
168    forall c : Components( #UAdmCtrl ); {
169      exists n : Components( #UQoSNeg ); {
170        exists pc : Ports( #c );
171               pn : Ports( #n ); {
172          exists s : Signature( #pn ); {
173            [Pipe( #c,#pc,#n,#pn ) in Pipes()] and
174            [PropertyValue( #s,“Type” ) == #PIntraLevel]
175          }
176        }
177      }
178    }
179  }
180
181  // Restrição de grafo conexo, utilizando pipes do tipo
182  // SignallingPipe, entre metacomponentes de negociação.
183  // SE o grafo de componentes é conexo, haverá sempre um conjunto
184  // com todos os componentes na configuração, ordenado do componente
185  // com menos associações ao componente com mais associações,
186  // que obedece a esse predicado.
187  FolPredicate FullyConnectedQoSNegs {
188    exists s : SequenceSet( Components( #UQoSNeg ) )
189                 where
190                   [Cardinality( #s ) > 1] and
191                   [Cardinality( #s ) ==
192                      Cardinality( Components( #UQoSNeg ) )]; {
193      forall i : { 1..Cardinality( #s )-1 }; {
194        exists j : { #i+1..Cardinality( #s ) }; {
195          exists pni : Ports( At( #s,#i ) );
196                 pnj : Ports( At( #s,#j ) ); {
197            [Pipe( At( #s,#i ),#pni,At( #s,#j ),#pnj )
198               in Pipes( #SignallingPipe )]
199          }
200        }
201      }
202    }
203  }
204}
```

10.4. Estilo `HierarchyNQoS`

```
1  Style HierarchyNQoS {
2    PipeType InterLevelPipe( impl,behv ) {
3      Implementation = #impl;
4      Behaviour = #behv;
5      Cardinality = { 1 .. };
6
7      AccessPoint in {
8        Cardinality = 1;
9        Type = #PInterLevel;
10       Direction = “in”;
11     }
12     AccessPoint out {
13       Cardinality = 1;
14       Type = #PInterLevel;
15       Direction = “out”;
16     }
17   }
18 }
```

11
Apêndice C

Este apêndice apresenta a especificação completa dos esquemas-base XML que compõem o núcleo de LindaX, bem como o esquema de extensão para a lógica FOL.

11.1.
Esquema XML `lindaxprop`

```
1  <xsd:schema
2    xmlns="http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
3    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4    targetNamespace=
5              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
6    elementFormDefault="qualified"
7    attributeFormDefault="qualified">
8
9    <xsd:annotation>
10     <xsd:documentation>
11       LindaX properties - Schema 1.0
12     </xsd:documentation>
13   </xsd:annotation>
14
15   <!--
16     TYPE: Property
17     A property is used to specify many things for vocabulary
18     elements, such as base- or meta-level indication, type indication,
19     etc.
20    -->
21   <xsd:complexType name="Property">
22     <xsd:sequence>
23       <xsd:element name= "name"
24                    type= "StringValue"/>
25       <xsd:element name= "value"
26                    type= "ElemExp"/>
27     </xsd:sequence>
28   </xsd:complexType>
29
30   <!--
31     TYPE: StringValue
32     A simple string is just a string constant.
33    -->
34   <xsd:simpleType name="StringValue">
35     <xsd:restriction base="xsd:string">
36     </xsd:restriction>
37   </xsd:simpleType>
38
39   <!--
40     TYPE: ElemExp
41     Represents a generic element expression, which can be
42     simple string, hexBinary, xml-link, set expression or a symbol.
43    -->
44   <xsd:complexType name="ElemExp">
45     <xsd:choice>
46       <xsd:element name="stringValue" type="StringValue"/>
```

```
47      <xsd:element name="integerValue" type="IntegerValue"/>
48      <xsd:element name="set" type="Set"/>
49      <xsd:element name="externalRef" type="ExternalRef"/>
50    </xsd:choice>
51  </xsd:complexType>
52
53  <!--
54    TYPE: IntegerValue
55
56    A simple integer in LindaX is just a
57    representation of an hexBinary.
58   -->
59  <xsd:simpleType name="IntegerValue">
60    <xsd:restriction base="xsd:hexBinary">
61    </xsd:restriction>
62  </xsd:simpleType>
63
64  <!--
65    TYPE: Set
66
67    A Set is a choice among different sets.
68   -->
69  <xsd:complexType name="Set">
70    <xsd:choice>
71      <xsd:element name="list" type="List"/>
72      <xsd:element name="range" type="Range"/>
73    </xsd:choice>
74  </xsd:complexType>
75
76  <!--
77    TYPE: List
78
79    A List is an enumeration.
80   -->
81  <xsd:complexType name="List">
82    <xsd:sequence>
83      <xsd:element name=     "elem"
84                   type=     "ElemExp"
85                   minOccurs="0"
86                   maxOccurs="unbounded" />
87    </xsd:sequence>
88  </xsd:complexType>
89
90  <!--
91    TYPE: Range
92
93    A Range is a range of numbers.
94   -->
95  <xsd:complexType name="Range">
96    <xsd:sequence>
97      <xsd:element name=     "lowElem"
98                   type=     "IntegerValue"/>
99      <xsd:element name=     "hiElem"
100                  type=     "IntegerValue"
101                  minOccurs="0"
102                  maxOccurs="1" />
103   </xsd:sequence>
104 </xsd:complexType>
105
106 <!--
107   TYPE: ExternalRef
108   An ExternalRef is an xml-link with parameters.
109  -->
110 <xsd:complexType name="ExternalRef">
111   <xsd:sequence>
112     <xsd:element name=     "externalSymbol"
113                  type=     "XMLLink"/>
114     <xsd:element name=     "externalParam"
115                  type=     "ExternalParam"
116                  minOccurs="0"
117                  maxOccurs="unbounded" />
118   </xsd:sequence>
119 </xsd:complexType>
```

```
120
121  <!--
122    TYPE: XMLLink
123
124    Encapsulates the parts of the XLink definition
125    that are useful in LindaX.
126    Imported from xArch definitions.
127   -->
128  <xsd:complexType name="XMLLink">
129    <xsd:attribute ref="xlink:type"/>
130    <xsd:attribute ref="xlink:href"/>
131  </xsd:complexType>
132
133  <!--
134    TYPE: ExternalParam
135   -->
136  <xsd:complexType name="ExternalParam">
137    <xsd:choice>
138      <xsd:element name=     "externalSymbol"
139                   type=     "XMLLink"/>
140      <xsd:element name=     "stringValue"
141                   type=     "StringValue"/>
142    </xsd:choice>
143  </xsd:complexType>
144
145  <!--
146    TYPE: Symbol
147
148    A symbol is a string representation of a variable which an
149    element can be linked to (eg: externalSymbol in <externalRef>).
150   -->
151  <xsd:complexType name="Symbol">
152    <xsd:attribute name=     "id"
153                   type=     "xsd:ID" />
154  </xsd:complexType>
155
156  <!--
157    TYPE: Parameter
158
159    A parameter is used to specify many things for vocabulary
160    elements externally, specially properties.
161   -->
162  <xsd:complexType name="Parameter">
163    <xsd:attribute name=     "id"
164                   type=     "xsd:ID" />
165  </xsd:complexType>
166
167</xsd:schema>
```


11.2. Esquema `lindaxtyp`

```
1  <xsd:schema
2    xmlns="http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
3    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4    xmlns:lindaxprop=
5              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
6    targetNamespace=
7               "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
8    elementFormDefault="qualified"
9    attributeFormDefault="qualified">
10
11   <xsd:import namespace=
12          "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
13               schemaLocation=
14          "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"/>
15
16   <xsd:annotation>
17     <xsd:documentation>
```

```
18      LindaX Type System - Schema 1.0
19    </xsd:documentation>
20  </xsd:annotation>
21
22  <!--
23    TYPE: Vocabulary
24
25    A vocabulary is an specification of a computation or
26    communication abstraction.
27    The specific format of a vocabulary is unspecified at this level.
28   -->
29  <xsd:complexType name="Vocabulary">
30    <xsd:sequence>
31      <xsd:element name=     "property"
32                   type=     "lindaxprop:Property"
33                   minOccurs="0"
34                   maxOccurs="unbounded" />
35    </xsd:sequence>
36    <xsd:attribute name=     "id"
37                   type=     "xsd:ID" />
38  </xsd:complexType>
39
40  <!--
41    TYPE: InterfaceType
42
43    The InterfaceType type defines a type of interface.
44    At the types level, no semantic information (such as a
45    behaviour) is defined in an interface type.
46    This can be specified in an extension or understood
47    programmatically.
48   -->
49  <xsd:complexType name="InterfaceType">
50    <xsd:complexContent>
51      <xsd:extension base="Vocabulary">
52        <xsd:sequence>
53          <xsd:element name=     "parameter"
54                       type=     "lindaxprop:Parameter"
55                       minOccurs="0"
56                       maxOccurs="unbounded" />
57        </xsd:sequence>
58      </xsd:extension>
59    </xsd:complexContent>
60  </xsd:complexType>
61
62  <!--
63    TYPE: ComponentType
64
65    The ComponentType type defines a type of component.  A type of
66    component is identified by its ID and ports.
67   -->
68  <xsd:complexType name="ComponentType">
69    <xsd:complexContent>
70      <xsd:extension base="lindaxstyle:Vocabulary">
71        <xsd:sequence>
72          <xsd:element name=     "parameter"
73                       type=     "lindaxprop:Parameter"
74                       minOccurs="0"
75                       maxOccurs="unbounded" />
76          <xsd:element name=     "port"
77                       type=     "Port"
78                       minOccurs="0"
79                       maxOccurs="unbounded" />
80        </xsd:sequence>
81      </xsd:extension>
82    </xsd:complexContent>
83  </xsd:complexType>
84
85  <!--    TYPE: PipeType
86
87    The PipeType type defines a type of pipe.  A type of
88    pipe is identified by its ID and access points.
89   -->
90  <xsd:complexType name="PipeType">
```

```
91     <xsd:complexContent>
92       <xsd:extension base="Vocabulary">
93         <xsd:sequence>
94           <xsd:element name=     "parameter"
95                        type=     "lindaxprop:Parameter"
96                        minOccurs="0"
97                        maxOccurs="unbounded" />
98           <xsd:element name=     "accessPoint"
99                        type=     "AccessPoint"
100                       minOccurs="0"
101                       maxOccurs="unbounded" />
102        </xsd:sequence>
103      </xsd:extension>
104    </xsd:complexContent>
105  </xsd:complexType>
106
107  <!--
108    TYPE: Port
109
110    The Port type describes an opaque port for use
111    at the structure level.  No semantic information is provided
112    at this level.  An opaque port contains an ID and signatures.
113    This may be connected to access points via an attachment.
114   -->
115  <xsd:complexType name="Port">
116    <xsd:complexContent>
117      <xsd:extension base="Vocabulary">
118        <xsd:sequence>
119          <xsd:element name=     "signature"
120                       type=     "Signature"
121                       minOccurs="0"
122                       maxOccurs="unbounded" />
123        </xsd:sequence>
124      </xsd:extension>
125    </xsd:complexContent>
126  </xsd:complexType>
127  <!--
128    TYPE: AccessPoint
129
130    The AccessPoint type describes an opaque access point for use
131    at the structure level.  No semantic information is provided
132    at this level.  An opaque access point contains an ID and
133    Signatures.  This may be connected to ports via an attachment.
134   -->
135  <xsd:complexType name="AccessPoint">
136    <xsd:complexContent>
137      <xsd:extension base="Vocabulary">
138        <xsd:sequence>
139          <xsd:element name=     "signature"
140                       type=     "Signature"
141                       minOccurs="0"
142                       maxOccurs="unbounded" />
143        </xsd:sequence>
144      </xsd:extension>
145    </xsd:complexContent>
146  </xsd:complexType>
147
148  <!--
149    TYPE: Signature
150
151    The Signature type defines one (of many)
152    "signatures" that a component or pipe type can possess.
153     A signature basically says, "FOO type
154    components/providers, when instantiated, should contain an
155    instance of BAR type interface.  The 'type' pointer is provided
156    as a property.
157   -->
158  <xsd:complexType name="Signature">
159    <xsd:complexContent>
160      <xsd:extension base="Vocabulary">
161      </xsd:extension>
162    </xsd:complexContent>
163  </xsd:complexType>
```

```
164
165</xsd:schema>
```

11.3. Esquema `lindaxcnf`

```
1  <xsd:schema
2    xmlns="http://www.telemidia.puc-rio.br/pub/LindaX/lindaxcnf.xsd"
3    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4    xmlns:lindaxprop=
5              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
6    xmlns:lindaxtyp=
7              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
8    targetNamespace=
9              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxcnf.xsd"
10   elementFormDefault="qualified"
11   attributeFormDefault="qualified">
12
13   <xsd:import namespace=
14         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
15        schemaLocation=
16         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd" />
17   <xsd:import namespace=
18         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
19        schemaLocation=
20         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd" />
21
22   <xsd:annotation>
23     <xsd:documentation>
24       LindaX Configuration Support - Schema 1.0
25     </xsd:documentation>
26   </xsd:annotation>
27
28   <!--
29     ELEMENT: lindaXConfiguration
30
31     The lindaXConfiguration element (of type Configuration)
32     is the root element that is the aegis over all other elements
33     in the configuration (either run- and design-time) structure.
34     This element is important for integration with xArch/ArchStudio.
35    -->
36   <xsd:element name="lindaXConfiguration" type="Configuration"/>
37
38
39   <!--
40     TYPE: Configuration
41
42     The Configuration type defines a structure for declaring
43     configurations (either run- and design-time) based on styles.
44    -->
45   <xsd:complexType name="Configuration">
46     <xsd:complexContent>
47       <xsd:extension base="lindaxtyp:Vocabulary">
48         <xsd:sequence>
49           <xsd:element name=     "parameter"
50                        type=     "lindaxprop:Parameter"
51                        minOccurs="0"
52                        maxOccurs="unbounded" />
53           <xsd:element name=     "instance"
54                        type=     "Instance"
55                        minOccurs="0"
56                        maxOccurs="unbounded" />
57           <xsd:element name=     "include"
58                        type=     "Include"
59                        minOccurs="0"
60                        maxOccurs="unbounded" />
61           <xsd:element name=     "pipe"
62                        type=     "Pipe"
63                        minOccurs="0"
```

```
                        maxOccurs="unbounded" />
          <xsd:element name=     "mapping"
                        type=     "Mapping"
                        minOccurs="0"
                        maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!--
    TYPE: System

    The System type defines a structure for declaring
    run-time configurations based on styles.
   -->
  <xsd:complexType name="System">
    <xsd:complexContent>
      <xsd:extension base="Configuration">
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!--
    TYPE: SystemTemplate

    The SystemTemplate type defines a structure for declaring
    design-time configurations based on styles.
   -->
  <xsd:complexType name="SystemTemplate">
    <xsd:complexContent>
      <xsd:extension base="Configuration">
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

 <!--
    TYPE: Instance
  -->
 <xsd:complexType name="Instance">
    <xsd:complexContent>
      <xsd:extension base="lindaxtyp:Vocabulary">
        <xsd:sequence>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
 </xsd:complexType>

 <!--
    TYPE: Include
  -->
 <xsd:complexType name="Include">
    <xsd:sequence>
          <xsd:element name=     "symbol"
                        type=     "lindaxprop:Symbol"
                        minOccurs="1"
                        maxOccurs="1" />
          <xsd:element name=     "externalRef"
                        type=     "lindaxprop:ExternalRef"
                        minOccurs="1"
                        maxOccurs="1" />
    </xsd:sequence>
 </xsd:complexType>

 <!--
    TYPE: Pipe
  -->
 <xsd:complexType name="Pipe">
    <xsd:complexContent>
      <xsd:extension base="lindaxtyp:Vocabulary">
        <xsd:sequence>
          <xsd:element name=     "peers"
                        type=     "lindaxprop:ExternalRef"
```

```
137                       minOccurs="2"
138                       maxOccurs="unbounded" />
139         </xsd:sequence>
140       </xsd:extension>
141     </xsd:complexContent>
142   </xsd:complexType>
143
144   <!--
145     TYPE: Mapping
146    -->
147   <xsd:complexType name="Mapping">
148     <xsd:sequence>
149           <xsd:element name=     "symbol"
150                       type=     "lindaxprop:Symbol"
151                       minOccurs="1"
152                       maxOccurs="1" />
153           <xsd:element name=     "externalRef"
154                       type=     "lindaxprop:ExternalRef"
155                       minOccurs="1"
156                       maxOccurs="1" />
157     </xsd:sequence>
158   </xsd:complexType>
159
160</xsd:schema>
```

11.4. Esquema lindaxres


```
1  <xsd:schema
2    xmlns="http://www.telemidia.puc-rio.br/pub/LindaX/lindaxres.xsd"
3    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4    xmlns:lindaxprop=
5              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
6    xmlns:lindaxtyp=
7              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
8    targetNamespace=
9              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxres.xsd"
10   elementFormDefault="qualified"
11   attributeFormDefault="qualified">
12   <xsd:import namespace=
13         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
14        schemaLocation=
15         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd" />
16   <xsd:import namespace=
17         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
18        schemaLocation=
19         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd" />
20
21   <xsd:annotation>
22     <xsd:documentation>
23       LindaX Resource View Support - Schema 1.0
24     </xsd:documentation>
25   </xsd:annotation>
26
27   <!--
28     ELEMENT: lindaXResource
29
30     The lindaXResource element (of type Resource)
31     is the root element that is the aegis over all other elements
32     in an resource view structure.
33     This element is important for integration with xArch/ArchStudio.
34   -->
35   <xsd:element name="lindaXResource" type="Resource"/>
36
37
38   <!--
39     TYPE: Resource
40
41     The Resource type defines a structure for declaring
```

```
42     resource views.
43    -->
44   <xsd:complexType name="Resource">
45     <xsd:complexContent>
46       <xsd:extension base="lindaxtyp:Vocabulary">
47         <xsd:sequence>
48           <xsd:element name=     "parameter"
49                        type=     "lindaxprop:Parameter"
50                        minOccurs="0"
51                        maxOccurs="unbounded" />
52         </xsd:sequence>
53       </xsd:extension>
54     </xsd:complexContent>
55   </xsd:complexType>
56
57   <!--
58     TYPE: Task
59
60     The Task type defines a structure for declaring
61     computational resource containers.
62    -->
63   <xsd:complexType name="Task">
64     <xsd:complexContent>
65       <xsd:extension base="Resource">
66       </xsd:extension>
67     </xsd:complexContent>
68   </xsd:complexType>
69
70   <!--
71     TYPE: Provider
72
73     The Provider type defines a structure for declaring
74     communication resource containers.
75    -->
76   <xsd:complexType name="Provider">
77     <xsd:complexContent>
78       <xsd:extension base="Resource">
79       </xsd:extension>
80     </xsd:complexContent>
81   </xsd:complexType>
```

11.5. Esquema `lindaxsty`

```
1  <xsd:schema
2    xmlns="http://www.telemidia.puc-rio.br/pub/LindaX/lindaxsty.xsd"
3    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4    xmlns:lindaxprop=
5              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
6    xmlns:lindaxtyp=
7              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
8    targetNamespace=
9              "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxsty.xsd"
10   elementFormDefault="qualified"
11   attributeFormDefault="qualified">
12
13   <xsd:import namespace=
14         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
15        schemaLocation=
16         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd" />
17   <xsd:import namespace=
18         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd"
19        schemaLocation=
20         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxtyp.xsd" />
21
22   <xsd:annotation>
23     <xsd:documentation>
24       LindaX styles - Schema 1.0
25     </xsd:documentation>
```

```
  </xsd:annotation>


  <!--
    ELEMENT: lindaXStyle

    The lindaXStyle element (of type Style) is the root element
    that is the aegis over all other elements in the style structure.
    This element is important for integration with xArch/ArchStudio.
   -->
  <xsd:element name="lindaXStyle" type="Style"/>

  <!--
    TYPE: Style

    The Style type describes the vocabulary and predicates that make
    part of an architectural style in LindaX.
    Styles at this level have no semantic information associated
    with them.  However, they give the designer and associated
    tools the ability to understand that an assembly of
    computation and communcation elements that conform to a
    particular set of predicates and vocabulary definitions.
    No specifications of what c&c elements actually do are provided
    at this level beyond the opaque descriptions associated with
    the vocabulary and predicates.
   -->
  <xsd:complexType name="Style">
    <xsd:sequence>
      <xsd:element name=     "superstyle"
                   type=     "lindaxprop:XMLLink"
                   minOccurs="0"
                   maxOccurs="1" />
      <xsd:element name=     "vocabulary"
                   type=     "lindaxtyp:Vocabulary"
                   minOccurs="0"
                   maxOccurs="unbounded" />
      <xsd:element name=     "predicate"
                   type=     "Predicate"
                   minOccurs="0"
                   maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name=     "id"
                   type=     "xsd:ID" />
  </xsd:complexType>

  <!--
    TYPE: Predicate

    A Predicate is an specification that must be satisfied
    by any assembly of components subject to the enclosing style.
    The specific format of a predicate is unspecified at this level.
   -->
  <xsd:complexType name="Predicate">
    <xsd:sequence>
      <xsd:element name=     "property"
                   type=     "lindaxprop:Property"
                   minOccurs="0"
                   maxOccurs="unbounded" />
      <xsd:element name=     "parameter"
                   type=     "lindaxprop:Parameter"
                   minOccurs="0"
                   maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name=     "id"
                   type=     "xsd:ID" />
  </xsd:complexType>

</xsd:schema>
```

11.6. Esquema lindaxfolpredicate

```
<xsd:schema
  xmlns=
   "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxfolpredicate.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lindaxprop=
            "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
  xmlns:lindaxsty=
             "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxsty.xsd"
  targetNamespace=
   "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxfolpredicate.xsd"
  elementFormDefault="qualified"
  attributeFormDefault="qualified">

  <xsd:import namespace=
        "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd"
             schemaLocation=
        "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxprop.xsd" />
  <xsd:import namespace=
         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxsty.xsd"
             schemaLocation=
         "http://www.telemidia.puc-rio.br/pub/LindaX/lindaxsty.xsd" />

  <xsd:annotation>
    <xsd:documentation>
      First-order predicates in LindaX styles - Schema 1.0
    </xsd:documentation>
  </xsd:annotation>


  <!--
    TYPE: FolPredicate

    A FolPredicate is an specification in first-order logic that
    must be satisfied by any assembly of components subject to
    the enclosing style.
   -->
  <xsd:complexType name="FolPredicate">
    <xsd:complexContent>
      <xsd:extension base="lindaxsty:Predicate">
        <xsd:sequence>
          <xsd:element name=     "logicalExp"
                       type=     "FolLogicalExp"
                       minOccurs="1"
                       maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!--
    TYPE: FolLogicalExp

    Type that specifies a logical expression in FOL.
   -->
  <xsd:complexType name="FolLogicalExp">
    <xsd:choice>
      <xsd:element name=     "forAll"
                   type=     "FolForAll"
                   minOccurs="1"
                   maxOccurs="1"/>
      <xsd:element name=     "exists"
                   type=     "FolExists"
                   minOccurs="1"
                   maxOccurs="1"/>
      <xsd:element name=     "boolExp"
                   type=     "FolBooleanExp"
                   minOccurs="1"
                   maxOccurs="1"/>
    </xsd:choice>
```

```
70  </xsd:complexType>
71
72
73  <!--
74    TYPE: FolForAll
75
76    Type that specifies a universal operator in FOL.
77   -->
78  <xsd:complexType name="FolForAll">
79    <xsd:sequence>
80      <xsd:element name="decl"
81                   type="FolFormalParam"
82                   minOccurs="1"
83                   maxOccurs="unbounded" />
84      <xsd:element name="logicalExp"
85                   type="FolLogicalExp"
86                   minOccurs="1"
87                   maxOccurs="1" />
88    </xsd:sequence>
89  </xsd:complexType>
90
91  <!--
92    TYPE: FolExists
93
94    Type that specifies an existencial operator in FOL.
95   -->
96  <xsd:complexType name="FolExists">
97    <xsd:sequence>
98      <xsd:element name="decl"
99                   type="FolFormalParam"
100                  minOccurs="1"
101                  maxOccurs="unbounded" />
102     <xsd:element name="logicalExp"
103                  type="FolLogicalExp"
104                  minOccurs="1"
105                  maxOccurs="1" />
106   </xsd:sequence>
107 </xsd:complexType>
108
109 <!--
110   TYPE: FolBooleanExp
111
112   Type that specifies a boolean expression in FOL.
113  -->
114 <xsd:complexType name="FolBooleanExp">
115   <xsd:choice>
116     <xsd:element name="and" type="FolAnd"/>
117     <xsd:element name="or" type="FolOr"/>
118     <xsd:element name="not" type="FolNot"/>
119     <xsd:element name="cmpExp" type="FolCmpExp"/>
120     <xsd:element name="boolParenExp" type="FolBoolParenExp"/>
121   </xsd:choice>
122 </xsd:complexType>
123
124 <!--
125   TYPE: FolFormalParam
126
127   Type that specifies a formal parameter in FOL.
128  -->
129 <xsd:complexType name="FolFormalParam">
130   <xsd:sequence>
131     <xsd:element name=     "symbol"
132                  type=     "lindaxprop:Symbol"
133                  minOccurs="1"
134                  maxOccurs="1"/>
135     <xsd:element name=     "set"
136                  type=     "FolSetExp"
137                  minOccurs="1"
138                  maxOccurs="1"/>
139     <xsd:element name="condition"
140                  type="FolBooleanExp"
141                  minOccurs="0"
142                  maxOccurs="1" />
```

```
143    </xsd:sequence>
144  </xsd:complexType>
145
146  <!--
147    TYPE: FolSetExp
148
149    Type that specifies a set expression in FOL.
150   -->
151  <xsd:complexType name="FolSetExp">
152    <xsd:choice>
153      <xsd:element name="union" type="FolUnion"/>
154      <xsd:element name="intersection" type="FolIntersection"/>
155      <xsd:element name="minus" type="FolMinus"/>
156      <xsd:element name="simpleSetValue" type="FolSimpleSetValue"/>
157      <xsd:element name="setParenExp" type="FolSetParenExp"/>
158      <xsd:element name="function" type="FolFunction"/>
159      <xsd:element name="externalSymbol" type="lindaxprop:XMLLink"/>
160    </xsd:choice>
161  </xsd:complexType>
162
163  <!--
164    TYPE: FolUnion
165   -->
166  <xsd:complexType name="FolUnion">
167    <xsd:sequence>
168      <xsd:element name="setExp1" type="FolSetExp"/>
169      <xsd:element name="setExp2" type="FolSetExp"/>
170    </xsd:sequence>
171  </xsd:complexType>
172
173  <!--
174    TYPE: FolIntersection
175   -->
176  <xsd:complexType name="FolIntersection">
177    <xsd:sequence>
178      <xsd:element name="setExp1" type="FolSetExp"/>
179      <xsd:element name="setExp2" type="FolSetExp"/>
180    </xsd:sequence>
181  </xsd:complexType>
182
183  <!--
184    TYPE: FolMinus
185   -->
186  <xsd:complexType name="FolMinus">
187    <xsd:sequence>
188      <xsd:element name="setExp1" type="FolSetExp"/>
189      <xsd:element name="setExp2" type="FolSetExp"/>
190    </xsd:sequence>
191  </xsd:complexType>
192
193  <!--
194    TYPE: FolSetParenExp
195
196    Type that specifies a set expression in parenthesis in FOL.
197   -->
198  <xsd:complexType name="FolSetParenExp">
199    <xsd:sequence>
200      <xsd:element name="setExp"
201                   type="FolSetExp"
202                   minOccurs="1"
203                   maxOccurs="1" />
204    </xsd:sequence>
205  </xsd:complexType>
206
207  <!--
208    TYPE: FolSimpleSetValue
209
210    A FolSet is a choice among different sets known in FOL.
211   -->
212  <xsd:complexType name="FolSimpleSetValue">
213    <xsd:choice>
214      <xsd:element name="list" type="FolList"/>
215      <xsd:element name="range" type="FolRange"/>
```

```
216    </xsd:choice>
217  </xsd:complexType>
218
219  <!--
220    TYPE: FolList
221
222    A FolList is an enumeration.
223   -->
224  <xsd:complexType name="FolList">
225    <xsd:sequence>
226      <xsd:element name=     "elem"
227                   type=     "FolElemExp"
228                   minOccurs="0"
229                   maxOccurs="unbounded" />
230    </xsd:sequence>
231  </xsd:complexType>
232
233  <!--
234    TYPE: FolRange
235
236    A FolRange is a range of numbers.
237   -->
238  <xsd:complexType name="FolRange">
239    <xsd:sequence>
240      <xsd:element name=     "lowElem"
241                   type=     "FolIntegerExp"
242                   minOccurs="0"
243                   maxOccurs="1" />
244      <xsd:element name=     "hiElem"
245                   type=     "FolIntegerExp"
246                   minOccurs="0"
247                   maxOccurs="1" />
248
249    </xsd:sequence>
250  </xsd:complexType>
251
252  <!--
253    TYPE: FolFunction
254
255    A FolFunction is a function that can be evaluated in the FOL.
256   -->
257  <xsd:complexType name="FolFunction">
258    <xsd:sequence>
259      <xsd:element name=     "name"
260                   type=     "lindaxprop:StringValue"
261                   minOccurs="1"
262                   maxOccurs="1" />
263      <xsd:element name=     "param"
264                   type=     "FolElemExp"
265                   minOccurs="0"
266                   maxOccurs="unbounded" />
267    </xsd:sequence>
268  </xsd:complexType>
269
270  <!--
271    TYPE: FolAnd
272
273    Represents a logical AND expression with its both sides
274    (subexpressions), using polish notation
275   -->
276  <xsd:complexType name="FolAnd">
277    <xsd:sequence>
278      <xsd:element name="booleanExp1" type="FolBooleanExp"/>
279      <xsd:element name="booleanExp2" type="FolBooleanExp"/>
280    </xsd:sequence>
281  </xsd:complexType>
282
283  <!--
284    TYPE: FolOr
285
286    Represents a logical OR expression with its both sides
287    (subexpressions), using polish notation
288   -->
```

```
289  <xsd:complexType name="FolOr">
290    <xsd:sequence>
291      <xsd:element name="booleanExp1" type="FolBooleanExp"/>
292      <xsd:element name="booleanExp2" type="FolBooleanExp"/>
293    </xsd:sequence>
294  </xsd:complexType>
295
296  <!--
297    TYPE: FolNot
298
299    Represents the unary logical NOT operation
300   -->
301  <xsd:complexType name="FolNot">
302    <xsd:sequence>
303      <xsd:element name="booleanExp" type="FolBooleanExp"/>
304    </xsd:sequence>
305  </xsd:complexType>
306
307  <!--
308    TYPE: FolBoolParenExp
309
310    Type that specifies a boolean expression in parenthesis in FOL.
311   -->
312  <xsd:complexType name="FolBoolParenExp">
313    <xsd:sequence>
314      <xsd:element name="boolExp"
315                   type="FolBooleanExp"
316                   minOccurs="1"
317                   maxOccurs="1" />
318    </xsd:sequence>
319  </xsd:complexType>
320
321  <!--
322    TYPE: FolCmpExp
323
324    Represents a comparison operation evaluating to TRUE or FALSE.
325   -->
326  <xsd:complexType name="FolCmpExp">
327    <xsd:choice>
328      <xsd:element name="equals"
329                   type="FolEquals"/>
330      <xsd:element name="notEquals"
331                   type="FolNotEquals"/>
332      <xsd:element name="greaterThanOrEquals"
333                   type="FolGreaterThanOrEquals"/>
334      <xsd:element name="lessThanOrEquals"
335                   type="FolLessThanOrEquals"/>
336      <xsd:element name="greaterThan"
337                   type="FolGreaterThan"/>
338      <xsd:element name="lessThan"
339                   type="FolLessThan"/>
340      <xsd:element name="inSet"
341                   type="FolInSet"/>
342    </xsd:choice>
343  </xsd:complexType>
344
345  <!--
346    TYPE: FolEquals
347
348    Represents a comparison operation evaluating to TRUE or FALSE.
349   -->
350  <xsd:complexType name="FolEquals">
351    <xsd:sequence>
352      <xsd:element name="elemExp1" type="FolElemExp"/>
353      <xsd:element name="elemExp2" type="FolElemExp"/>
354    </xsd:sequence>
355  </xsd:complexType>
356
357  <!--
358    TYPE: FolNotEquals
359
360    Represents a comparison operation evaluating to TRUE or FALSE.
361   -->
```

```
362  <xsd:complexType name="FolNotEquals">
363    <xsd:sequence>
364      <xsd:element name="elemExp1" type="FolElemExp"/>
365      <xsd:element name="elemExp2" type="FolElemExp"/>
366    </xsd:sequence>
367  </xsd:complexType>
368
369  <!--
370    TYPE: FolGreaterThanOrEquals
371
372    Represents a comparison operation evaluating to TRUE or FALSE.
373   -->
374  <xsd:complexType name="FolGreaterThanOrEquals">
375    <xsd:sequence>
376      <xsd:element name="elemExp1" type="FolElemExp"/>
377      <xsd:element name="elemExp2" type="FolElemExp"/>
378    </xsd:sequence>
379  </xsd:complexType>
380
381  <!--
382    TYPE: FolLessThanOrEquals
383
384    Represents a comparison operation evaluating to TRUE or FALSE.
385   -->
386  <xsd:complexType name="FolLessThanOrEquals">
387    <xsd:sequence>
388      <xsd:element name="elemExp1" type="FolElemExp"/>
389      <xsd:element name="elemExp2" type="FolElemExp"/>
390    </xsd:sequence>
391  </xsd:complexType>
392
393  <!--
394    TYPE: FolGreaterThan
395
396    Represents a comparison operation evaluating to TRUE or FALSE.
397   -->
398  <xsd:complexType name="FolGreaterThan">
399    <xsd:sequence>
400      <xsd:element name="elemExp1" type="FolElemExp"/>
401      <xsd:element name="elemExp2" type="FolElemExp"/>
402    </xsd:sequence>
403  </xsd:complexType>
404
405  <!--
406    TYPE: FolLessThan
407
408    Represents a comparison operation evaluating to TRUE or FALSE.
409   -->
410  <xsd:complexType name="FolLessThan">
411    <xsd:sequence>
412      <xsd:element name="elemExp1" type="FolElemExp"/>
413      <xsd:element name="elemExp2" type="FolElemExp"/>
414    </xsd:sequence>
415  </xsd:complexType>
416
417  <!--
418    TYPE: FolInSet
419
420    Represents a comparison operation evaluating to TRUE or FALSE.
421   -->
422  <xsd:complexType name="FolInSet">
423    <xsd:sequence>
424      <xsd:element name="elemExp" type="FolElemExp"/>
425      <xsd:element name="setExp" type="FolSetExp"/>
426    </xsd:sequence>
427  </xsd:complexType>
428
429  <!--
430    TYPE: FolElemExp
431
432    Represents a generic element expression, which can be
433    simple string, integer expression, set expression,
434    function expression or a symbol.
```

```
435   -->
436  <xsd:complexType name="FolElemExp">
437    <xsd:choice>
438      <xsd:element name="simpleStr"
439                   type="lindaxprop:StringValue"/>
440      <xsd:element name="externalSymbol"
441                   type="lindaxprop:XMLLink"/>
442      <xsd:element name="function"
443                   type="FolFunction"/>
444      <xsd:element name="set"
445                   type="FolSetExp"/>
446      <xsd:element name="intExp"
447                   type="FolIntegerExp"/>
448    </xsd:choice>
449  </xsd:complexType>
450
451  <!--
452    TYPE: FolIntegerExp
453
454    Type that specifies an integer expression in FOL.
455   -->
456  <xsd:complexType name="FolIntegerExp">
457    <xsd:choice>
458      <xsd:element name="simpleInt"
459                   type="lindaxprop:IntegerValue"/>
460      <xsd:element name="externalSymbol"
461                   type="lindaxprop:XMLLink"/>
462      <xsd:element name="function"
463                   type="FolFunction"/>
464      <xsd:element name="add"
465                   type="FolAdd"/>
466      <xsd:element name="sub"
467                   type="FolSub"/>
468      <xsd:element name="mult"
469                   type="FolMult"/>
470      <xsd:element name="div"
471                   type="FolDiv"/>
472      <xsd:element name="intParenExp"
473                   type="FolIntParenExp"/>
474    </xsd:choice>
475  </xsd:complexType>
476
477  <!--
478    TYPE: FolAdd
479
480    Represents an integer sum operation.
481   -->
482  <xsd:complexType name="FolAdd">
483    <xsd:sequence>
484      <xsd:element name="intExp1" type="FolIntegerExp"/>
485      <xsd:element name="intExp2" type="FolIntegerExp"/>
486    </xsd:sequence>
487  </xsd:complexType>
488
489  <!--
490    TYPE: FolSub
491
492    Represents an integer subtraction operation.
493   -->
494  <xsd:complexType name="FolSub">
495    <xsd:sequence>
496      <xsd:element name="intExp1" type="FolIntegerExp"/>
497      <xsd:element name="intExp2" type="FolIntegerExp"/>
498    </xsd:sequence>
499  </xsd:complexType>
500
501  <!--
502    TYPE: FolMult
503
504    Represents an integer multiplication operation.
505   -->
506  <xsd:complexType name="FolMult">
507    <xsd:sequence>
```

```
508      <xsd:element name="intExp1" type="FolIntegerExp"/>
509      <xsd:element name="intExp2" type="FolIntegerExp"/>
510    </xsd:sequence>
511  </xsd:complexType>
512
513  <!--
514    TYPE: FolDiv
515
516    Represents an integer division operation.
517   -->
518  <xsd:complexType name="FolDiv">
519    <xsd:sequence>
520      <xsd:element name="intExp1" type="FolIntegerExp"/>
521      <xsd:element name="intExp2" type="FolIntegerExp"/>
522    </xsd:sequence>
523  </xsd:complexType>
524
525  <!--
526    TYPE: FolIntParenExp
527
528    Represents an integer parenthesised expression.
529   -->
530  <xsd:complexType name="FolIntParenExp">
531    <xsd:sequence>
532      <xsd:element name="intExp" type="FolIntegerExp"/>
533    </xsd:sequence>
534  </xsd:complexType>
535
536</xsd:schema>
```