8
Referências Bibliográficas

ABADI, M. et al. Dynamic typing in a statically typed language. **ACM Transactions on Programming Languages and Systems**, ACM Press, Nova Iorque, EUA, v. 13, n. 2, p. 237–268, 1991. 4.5, 7.2

AJMANI, S.; LISKOV, B.; SHRIRA, L. Modular software upgrades for distributed systems. In: THOMAS, D. (Ed.). **Anais do ECOOP'06**. Berlin, Alemanha: Springer, 2006. (Lecture Notes in Computer Science, v. 4067), p. 452–476. ISBN 3-540-35726-2. 7.2

AKTEMUR, B. et al. Optimizing marshalling by run-time program generation. In: GLÜCK, R.; LOWRY, M. R. (Ed.). **Anais do GPCE 2005**. Nova Iorque, EUA: Springer, 2005. (Lecture Notes in Computer Science, v. 3676), p. 221–236. ISBN 3-540-29138-5. 3.5.2

ARULANTHU, A. B. et al. Applying patterns and components to develop an IDL compiler for CORBA AMI callbacks. **C++ Report**, v. 12, n. 3, mar. 2000. 4.3.2

BASIC. Hanover, EUA, 1964. Disponível em: <`http://www.bitsavers.org/pdf/dartmouth/BASIC_Oct64.pdf`>. Acesso em: mar., 2009. 1

BATISTA, T.; VIEIRA, M. RE-AspectLua - achieving reuse in AspectLua. **Journal of Universal Computer Science**, v. 13, n. 6, p. 786–805, jun. 2007. 2.2.1, 4.4.3

BIRRELL, A.; NELSON, B. J. Implementing remote procedure calls. **ACM Trans. Comput. Syst.**, v. 2, n. 1, p. 39–59, 1984. 2.1

BLACKWELL, A. F. et al. Cognitive dimensions of notations: Design tools for cognitive technology. In: BEYNON, M.; NEHANIV, C. L.; DAUTENHAHN, K. (Ed.). **Anais do Cognitive Technology 2001**. Berlin, Alemanha: Springer, 2001. (Lecture Notes in Computer Science, v. 2117), p. 325–341. 2.4

BLACKWELL, A. F.; GREEN, T. R. G. A cognitive dimensions questionnaire optimised for users. In: BLACKWELL, A. F.; BILOTTA, E. (Ed.). **Anais do**

PPIG'00. Cosenza, Itália: Psychology of Programming Interest Group, 2000. p. 137–154. 7.2

BLAIR, G. S. et al. An architecture for next generation middleware. In: **Middleware, IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98), Lake District**. Alemanha: Springer, 1998. p. 15–18. 6.1

BOCTOR, D. **Microsoft Office 2000 Visual Basic for applications : Fundamentals**. Redmond, EUA: Microsoft Press, 1999. 2.2

BOX, D. **Essential COM**. Boston, EUA: Addison-Wesley Longman, 1997. ISBN 0201634465. 6.1.3

BOX, D. et al. **Simple Object Access Protocol (SOAP) 1.1**. Cambridge, EUA, maio 2000. Disponível em: <`http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`>. Acesso em: abr., 2009. 7.3

BROSE, G. JacORB: Implementation and design of a Java-ORB. In: KÖNIG, H.; GEIHS, K.; PREUSS, T. (Ed.). **Anais do DAIS'97**. Londres, Reino Unido: Chapman & Hall, 1997. p. 143–154. 1

CACHO, N. et al. Aspect Open-ORB: um middleware reflexivo orientado a aspectos. In: GRANVILLE, L. Z. et al. (Ed.). **Anais do SBRC'07**. Belém, Brasil: Universidade Federal do Pará, 2007. p. 1079–1084. 6.1.3

CARZANIGA, A.; ROSENBLUM, D. S.; WOLF, A. L. Achieving scalability and expressiveness in an internet-scale event notification service. In: NEIGER, G. (Ed.). **Anais do PODC'00**. Nova Iorque, EUA: ACM Press, 2000. p. 219–227. ISBN 1-58113-183-6. 1, 2.1

CASTOR FILHO, F. et al. A group membership service for large-scale grids. In: **MGC '08: Proceedings of the 6th international workshop on Middleware for grid computing**. Nova Iorque, EUA: ACM Press, 2008. p. 1–6. ISBN 978-1-60558-365-5. 7.1

CERQUEIRA, R. F. de G.; CASSINO, C.; IERUSALIMSCHY, R. Dynamic component gluing across different componentware systems. In: TARI, Z. et al. (Ed.). **Anais do DOA'99**. Washington, EUA: IEEE Computer Society Press, 1999. p. 362–373. 3, 6.2.1, 7.3, 7.3

CLARKE, S. Evaluating a new programming language. In: KADODA, G. F. (Ed.). **13th Annual Workshop of the Psychology of Programming**

Interest Group. Keele, Reino Unido: Universidade de Keele, 2001. p. 275–289. 1, 2.4

CLARKE, S. Describing and measuring API usability with the Cognitive Dimensions. In: **Cognitive Dimensions of Notations 10th Anniversary Workshop**. [s.n.], 2006. Disponível em: <`http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/workshop2005/Clarke_position_paper.pdf`>. Acesso em: mar., 2009. 1, 2.4

CLARKE, S.; BECKER, C. Using the Cognitive Dimensions Framework to evaluate the usability of a class library. In: PETRE, M.; BUDGEN, D. (Ed.). **15th Annual Workshop of the Psychology of Programming Interest Group**. Keele, Reino Unido: Universidade de Keele, 2003. p. 359–366. 1

COLGAN, L. MATLAB in first-year engineering mathematics. **International Journal of Mathematical Education in Science and Technology**, v. 31, n. 1, p. 15–25, jan. 2000. 2.4.12

CORREA, S. **SCS: Software Component System**. maio 2007. Disponível em: <`http://www.tecgraf.puc-rio.br/~scorrea/scs`>. Acesso em: dez., 2007. 7.1

COSTA, F. M. et al. The role of reflective middleware in supporting the engineering of dynamic applications. In: CAZZOLA, W.; STROUD, R. J.; TISATO, F. (Ed.). **Reflection and Software Engineering**. Nova Iorque, EUA: Springer, 2000. (Lecture Notes in Computer Science, v. 1826), p. 79–99. 6.1.3

COSTA, P. et al. The RUNES middleware: A reconfigurable component-based approach to networked embedded systems. In: **Anais do PIMRC '05**. Nova Iorque, EUA: IEEE Computer Society Press, 2005. v. 2, p. 806–810. 6.1.3, 7.3

COULSON, G. et al. The design of a configurable and reconfigurable middleware platform. **Distributed Computing**, Springer, Londres, Reino Unido, v. 15, n. 2, p. 109–126, 2002. ISSN 0178-2770. 2.1, 6.1.3, 7.3

COULSON, G. et al. A component model for building systems software. In: HAMZA, M. H. (Ed.). **Anais do IASTED SEA'04**. Calgary, Canadá: ACTA Press, 2004. p. 684–689. 6.1.3

COULSON, G. et al. A generic component model for building systems software. **ACM Transactions on Computer Systems**, ACM Press, Nova Iorque, EUA, v. 26, n. 1, p. 1–42, 2008. 6.1.3, 7.3

COULSON, G. et al. A component-based middleware framework for configurable and reconfigurable grid computing: Research articles. **Concurrent Computing : Practice Experience**, John Wiley & Sons, v. 18, n. 8, p. 865–874, 2006. 6.1.3, 7.3

EDEN, A. H.; MENS, T. Measuring software flexibility. **IEE Software**, v. 153, n. 3, p. 113–126, jun. 2006. ISSN 1462-5970. 2.2.1, 2.3.1, 7.2

ELIASSEN, F. et al. Next generation middleware: Requirements, architecture, and prototypes. In: **Anais do FTDCS'99**. Los Alamitos, EUA: IEEE Computer Society Press, 1999. p. 60–68. ISBN 0-7695-0468-X. 2.1

ELLIOTT, C. **opalORB: A simple Perl CORBA ORB**. 2008. Disponível em: <`http://opalorb.sourceforge.net/`>. Acesso em: jan., 2008. 6.2.1

ELLIS, B.; STYLOS, J.; MYERS, B. A. The factory pattern in API design: A usability evaluation. In: **29th International Conference on Software Engineering**. Minneapolis, EUA: IEEE Computer Society Press, 2007. p. 302–312. 7.2

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Ph.D.) — Universidade da Califórnia, Irvine, EUA, 2000. 2.1

FLEUTOT, F.; TRATT, L. Contrasting compile-time meta-programming in Metalua and converge. In: **Workshop on Dynamic Languages and Applications**. [S.l.: s.n.], 2007. 4.4.4

FOWLER, M. **Inversion of Control Containers and the Dependency Injection pattern**. jan. 2008. Disponível em: <`http://www.martinfowler.com/articles/injection.html`>. Acesso em: abr., 2008. 3.2

FUSCO, V. S. F. **Travessia de Firewall/NAT usando CORBA**. Rio de Janeiro, Brasil, nov. 2007. 3.6

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Nova Iorque, EUA: Addison-Wesley Longman, 1994. Hardcover. ISBN 0201633612. 2.4.4, 3.2, 4.2.2

GARRETT, J. J. **AJAX: A New Approach to Web Applications**. fev. 2005. Disponível em: <`http://adaptivepath.com/ideas/essays/archives/000385.php`>. Acesso em: fev., 2009. 1, 2.1

GEELAN, J. James Gosling: "Java is under no serious threat from PHP, Ruby or C#". **JAVA Developer's Journal**, SYS-CON Media, maio 2007. Disponível em: <http://java.sys-con.com/node/193146>. Acesso em: mar., 2009. 1

GILL, C. D. et al. Applying adaptive real-time middleware to address grand challenges of COTS-based mission-critical real-time systems. In: **Anais do 1st International Workshop on Real-Time Mission-Critical Systems: Grand Challenge Problems**. Phoenix, EUA: IEEE Computer Society Press, 1999. 6.1.2

GOLDCHLEGER, A. et al. InteGrade: Object-oriented grid middleware leveraging idle computing power of desktop machines. **Concurrency and Computation: Practice and Experience**, v. 16, p. 449–459, mar. 2004. 7.1

GOSLING, J.; MCGILTON, H. **The Java^{TM} Language Environment**. Mountain View, EUA, maio 1996. 1

GRACE, P.; BLAIR, G. S.; SAMUEL, S. A reflective framework for discovery and interaction in heterogeneous mobile environments. **Mobile Computing and Communications Review**, v. 9, n. 1, p. 2–14, 2005. 6.1.3, 7.3

GREEN, T. R. G. Cognitive dimensions of notations. In: SUTCLIFFE, A.; MACAULAY, L. (Ed.). **People and Computers V**. Cambridge, Reino Unido: Cambridge University Press, 1989. (British Computer Society Workshop Series), p. 443–460. 1, 2.4, 7.2

GREEN, T. R. G.; BLACKWELL, A. F. **Cognitive Dimensions of Information Artefacts: A Tutorial**. out. 1998. Disponível em: <`http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf`>. Acesso em: mar., 2009. 2.3.1

GREEN, T. R. G.; PETRE, M. Usability analysis of visual programming environments: a 'Cognitive Dimensions' framework. **Journal of Visual Languages and Computing**, Elsevier Science, v. 7, n. 2, p. 131–174, jun. 1996. 2.4

GREENES, R. A. et al. A system for clinical data management. In: **AFIPS '69 (Fall): Proceedings of the November 18-20, 1969, fall joint computer conference**. Nova Iorque, EUA: ACM Press, 1969. p. 297–305. 1

GRISBY, D. **Free High Performance ORB**. out. 2008. Disponível em: <`http://omniorb.sourceforge.net/`>. Acesso em: mar., 2009. 6.2.1, 7.3, 7.3

GROSSO, W. **Java RMI**. Sebastopol, EUA: O'Reilly & Associates, 2001. 1, 2.1

GROUP, N. G. M. **OpenCOM: Java versus C++**. Lancaster, Reino Unido, 2007. Disponível em: <`http://www.comp.lancs.ac.uk/computing/research/mpg/reflection/opencomj.pdf`>. Acesso em: fev., 2009. 1, 2.1, 6.1.3

HAMMER. **static: The Multipurpose Keyword**. Disponível em: <`http://www.cprogramming.com/tutorial/statickeyword.html`>. Acesso em: mar., 2009. 2.4.13

HENNING, M. A new approach to object-oriented middleware. **IEEE Internet Computing**, v. 8, n. 1, p. 66–75, jan. 2004. 2.1, 7.3

HENNING, M. The rise and fall of CORBA. **Queue**, ACM Press, Nova Iorque, EUA, v. 4, n. 5, p. 28–34, jun. 2006. ISSN 1542-7730. 1

HIRSCHI, A. Traveling light, the Lua way. **IEEE Software**, IEEE Computer Society Press, Los Alamitos, EUA, v. 24, n. 5, p. 31–38, set. 2007. ISSN 0740-7459. 1, 2.2.1

IEEE STANDARD BOARD. **IEEE Standard Glossary of Software Engineering Terminology**. Std 610.12-1990. Annapolis, EUA, jan. 1991. 2.3

IERUSALIMSCHY, R. **Programming in Lua**. Segunda edição. Rio de Janeiro, Brasil: Lua.org, 2006. ISBN 8590379825. 1, 2.2.1

JAIN, P.; SCHMIDT, D. C. Dynamically configuring communication services with the service configurator pattern. **C++ Report**, v. 9, n. 6, 1997. 6.1.2

JONG, I. de. **Pyro - Python Remote Objects**. maio 2007. Disponível em: <`http://pyro.sourceforge.net/`>. Acesso em: jan., 2008. 1, 6.2.2

KON, F.; CAMPBELL, R. H. Supporting automatic configuration of component-based distributed systems. In: DEVARAKONDA, M. (Ed.). **Anais do COOTS'99**. São Diego, EUA: USENIX Association, 1999. p. 175–187. 2.1

KON, F.; CAMPBELL, R. H. Dependence management in component-based distributed systems. **IEEE Concurrency**, v. 8, p. 26–36, 2000. 6.1.2, 7.3

KON, F. et al. The case for reflective middleware. **Communications of ACM**, ACM Press, Nova Iorque, EUA, v. 45, n. 6, p. 33–38, jun. 2002. 1, 6.1.2

LIMA, M. J. et al. CSBase: A framework for building customized grid environments. In: **Anais do WETICE '06**. Washington, EUA: IEEE Computer Society Press, 2006. p. 187–194. ISBN 0-7695-2623-3. 7.1

MAIA, R.; CERQUEIRA, R. F. de G.; CALHEIROS, R. OiL: An object request broker in the Lua language. In: BATISTA, T. (Ed.). **Anais do SBRC'06 - Salão de Ferramentas**. Porto Alegre, Brasil: SBC, 2006. p. 1439–1446. 1

MAIA, R.; CERQUEIRA, R. F. de G.; KON, F. A middleware for experimentation on dynamic adaptation. In: **Anais do ARM'05 workshop**. Nova Iorque, EUA: ACM Press, 2005. 7.2

MAIA, R.; CERQUEIRA, R. F. de G.; RODRIGUEZ, N. An infrastructure for development of dynamically adaptable distributed components. In: MEERSMAN, R.; TARI, Z. (Ed.). **Anais do DOA'04**. Berlin, Alemanha: Springer, 2004. (Lecture Notes in Computer Science, v. 3291), p. 1285–1302. 3

MELLO, R. X. de. **Um Modelo de Escalonamento Colaborativo de Eventos Baseado em Corrotinas**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, set. 2005. 7.1

MILNER, R. A theory of type polymorphism in programming. **Journal of Computer and System Sciences**, v. 17, p. 348–375, 1978. 2.4.1

MONTANARI, R.; LUPU, E.; STEFANELLI, C. Policy-based dynamic reconfiguration of mobile-code applications. **Computer**, IEEE Computer Society Press, v. 37, n. 7, p. 73–80, jul. 2004. 1

MOURA, A. L. de; IERUSALIMSCHY, R. **Revisiting coroutines**. Rio de Janeiro, Brasil, jun. 2004. 3.3.4

MOURA, A. L. de et al. Dynamic support for distributed auto-adaptive applications. In: WAGNER, R. (Ed.). **Anais do ICDCS 2002 Workshops**. Washington, EUA: IEEE Computer Society Press, 2002. p. 451–458. ISBN 0-7695-1588-6. 3

MURPHY, G. C.; KERSTEN, M.; FINDLATER, L. How are Java software developers using the Eclipse IDE? **IEEE Software**, v. 23, n. 4, p. 76–83, jul. 2006. 2.4.10

MYERS, G. J. **The Art of Software Testing**. Nova Iorque, EUA: John Wiley & Sons, 1979. 2.4.6

NAUR, P.; RANDELL, B. (Ed.). **Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO**. Bruxelas, Bélgica: Scientific Affairs Division, NATO, 1968. 1

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: CHEW, J. C.; WHITESIDE, J. (Ed.). **Anais do CHI '90**. Nova Iorque, EUA: ACM Press, 1990. p. 249–256. 7.2

NOGARA, L. G. C. **Um Estudo Sobre Middlewares Adaptáveis**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, set. 2006. 3.6, 7.1

OBJECT MANAGEMENT GROUP. **Common Object Request Broker Architecture: Core Specification - Version 3.0**. Needham, EUA, dez. 2002. Document: formal/2002-12-06. 1, 2.1, 3.1, 3.5

OBJECT MANAGEMENT GROUP. **Python Language Mapping Specification**. Needham, EUA, nov. 2002. Document: formal/2002-11-05. 6.2.1, 7.3

OBJECT MANAGEMENT GROUP. **Ruby CORBA Language Mapping Specification**. Ftf beta 1. Needham, EUA, nov. 2009. Document: ptc/2009-01-01. 6.2.1, 7.3

OLIVEIRA NETO, J. J. d. Mestrado, **Uso de um Modelo de Interceptadores para Prover Adaptação Dinâmica no InteGrade**. Goiânia, Brasil: [s.n.], set. 2008. 3.6, 7.1

O'RYAN, C. et al. The design and performance of a pluggable protocols framework for real-time distributed object computing middleware. In: SVENTEK, J. S.; COULSON, G. (Ed.). **Anais do Middleware 2000**. Nova Iorque, EUA: Springer, 2000. (Lecture Notes in Computer Science, v. 1795), p. 372–395. ISBN 3-540-67352-0. 6.1.2

OUSTERHOUT, J. K. Scripting: Higher-level programming for the 21^{st} century. **Computer**, v. 31, n. 3, p. 23–30, mar. 1998. 2.4.12

PARLAVANTZAS, N. et al. Towards a reflective component-based middleware architecture. In: CAZZOLA, W.; CHIBA, S.; LEDOUX, T. (Ed.). **Anais do ECOOP'00 Workshops**. [s.n.], 2000. Disponível em: <`http://www.disi.unige.it/person/CazzolaW/ewrma2000-proceedings.html`>. Acesso em: fev., 2008. 6.1.3

PELLERIN, R. et al. GASP: an open source gaming service middleware dedicated to multiplayer games for J2ME based mobile phones. In: MEHDI, Q.; GOUGH, N.; NATKIN, S. (Ed.). **Anais do CGAME'05**. Annapolis, EUA: IEEE Press, 2005. 1

PIETZUCH, P. R.; BACON, J. M. Hermes: A distributed event-based middleware architecture. In: WAGNER, R. (Ed.). **Anais do ICDCS 2002 Workshops**. Washington, EUA: IEEE Computer Society Press, 2002. p. 611–618. ISBN 0-7695-1588-6. 2.1

PUDER, A.; RÖMER, K.; PILHOFER, F. **Distributed Systems Architecture: A Middleware Approach**. São Francisco, EUA: Elsevier Science, 2006. 1, 4, 4.1, 6.1.1, 7.1

QUARESMA, B.; MAIA, R.; GUISASOLA, T. **MPA - Módulo de Procedimentos Automatizados**. Rio de Janeiro, Brasil, nov. 2008. 7.1

RIEMAN, J.; FRANZKE, M.; REDMILES, D. Usability evaluation with the cognitive walkthrough. In: KATZ, I. R.; MACK, R. L.; MARKS, L. (Ed.). **Proceedings companion of CHI '95**. Nova Iorque, EUA: ACM Press, 1995. p. 387–388. 7.2

ROMÁN, M.; KON, F.; CAMPBELL, R. H. Reflective middleware: from your desk to your hand. **IEEE Distributed Systems Online**, v. 2, n. 5, 2001. 6.1.4

ROMÁN, M. et al. LegORB and ubiquitous CORBA. In: BLAIR, G. S.; CAMPBELL, R. H. (Ed.). **Anais do RM'00 workshop**. Nova Iorque, EUA: ACM Press, 2000. p. 1–2. 6.1.4

SCHENTAG, A. **Using the Set Command**. Disponível em: <`http://www.programmers-corner.com/viewTutorial.php/1`>. Acesso em: mar., 2009. 2.4.13

SCHMIDT, D. C. The ADAPTIVE communication environment: Object-oriented network programming components for developing client/server applications. In: **12th Sun Users Group Conference**. [S.l.: s.n.], 1994. 6.1.2

SCHMIDT, D. C. **Research on Real-time CORBA**. set. 2006. Disponível em: <`http://www.cs.wustl.edu/~schmidt/corba-research-realtime.html`>. Acesso em: mar., 2009. 6.1.2

SCHMIDT, D. C. **Real-time CORBA with TAO (The ACE ORB)**. jun. 2007. Disponível em: <`http://www.cse.wustl.edu/~schmidt/TAO.html`>. Acesso em: mar., 2009. 6.1.2

SCHMIDT, D. C.; LEVINE, D. L.; MUNGEE, S. The design of the TAO real-time object request broker. **Computer Communications**, Elsevier Science, v. 21, n. 4, p. 294–324, abr. 1998. 6.1.2

SCHMIDT, D. C. et al. **Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects**. Oxford, Reino Unido: John Wiley & Sons, 2000. 1, 4.2.2

SCHMIDT, D. C.; VINOSKI, S. The history of the OMG C++ mapping. **C/C++ Users Journal**, nov. 2000. 4.2.2

SIEK, J.; TAHA, W. Gradual typing for objects. In: **ECOOP '07: Proceedings of the 21st European conference on ECOOP 2007**. Berlin, Alemanha: Springer, 2007. p. 2–27. ISBN 978-3-540-73588-5. 4.5

SILVA, J. D. S. da. **AO-OIL - Um Middleware Reflexivo e Orientado a Aspectos baseado em uma Arquitetura de Referência**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, Brasil, fev. 2009. 7.1, 7.3

SOARES, L. F. G.; RODRIGUEZ, R. F.; MORENO, M. F. Ginga-NCL: the declarative environment of the brazilian digital TV system. **Journal of Brazilian Computer Society**, v. 13, p. 37–46, mar. 2007. 1

STOTHARD, P. The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences. **BioTechniques**, v. 28, n. 6, p. 1102–1104, jun. 2000. 1

SZYPERSKI, C. **Component Software: Beyond Object-Oriented Programming**. Nova Iorque, EUA: Addison-Wesley Longman, 1998. 3.2

TACKABERRY, J. **CORBA Objects in Python**. Dissertação (Mestrado) — Universidade de Algoma, Sault Ste. Marie, Canadá, abr. 2000. 6.2.1, 7.3, 7.3

TANENBAUM, A. S. **Modern Operating Systems**. Segunda edição. Upper Saddle River, EUA: Prentice Hall, 2001. 3.3.4

THEOPHILO, A.; ENDLER, M.; CERQUEIRA, R. F. de G. Evaluation of three approaches for CORBA firewall/NAT traversal. In: MEERSMAN, R.; TARI, Z. (Ed.). **Anais do DOA'05**. Berlin, Alemanha: Springer, 2005. (Lecture Notes in Computer Science, v. 3761), p. 923–940. 3.6, 7.1

THOMSON, D.; SMITH, R. **Fnorb: the pure Python CORBA ORB**. abr. 2005. Disponível em: <`http://fnorb.sourceforge.net/`>. Acesso em: jan., 2008. 6.2.1

TŮMA, P. **CPPsuite**. maio 2002. Disponível em: <`http://dsrg.mff.cuni.cz/~ceres/prj/CCPsuite/`>. Acesso em: abr., 2008. 7.3

PETRE, M.; BUDGEN, D. (Ed.). **15th Annual Workshop of the Psychology of Programming Interest Group**. Universidade de Keele, Keele, Reino Unido: Universidade de Keele, 2003.

URURAHY, C.; RODRIGUEZ, N.; IERUSALIMSCHY, R. ALua: Flexibility for parallel programming. **Computer Languages**, Elsevier Science, v. 28, n. 2, p. 155–180, 2002. 3

VINOSKI, S. Middleware "dark matter". **IEEE Internet Computing**, v. 6, n. 5, p. 92–95, set. 2002. 1

VINOSKI, S. The performance presumption. **IEEE Internet Computing**, v. 7, n. 2, p. 88–90, mar. 2003. 1

VINOSKI, S. The language divide. **IEEE Internet Computing**, v. 10, n. 2, p. 82–84, mar. 2006. 2.1

WAGNER, R. (Ed.). **22nd International Conference on Distributed Computing Systems, Workshops (ICDCSW '02) July 2-5, 2002, Vienna, Austria, Proceedings**. Washington, EUA: IEEE Computer Society Press, 2002. ISBN 0-7695-1588-6.

WHITEHEAD II, J.; MCLEMORE, B.; ORLANDO, M. **World of Warcraft Programming: A Guide and Reference for Creating WoW Addons**. Edição ilustrada. Oxford, Reino Unido: John Wiley & Sons, 2008. 2.2

YEGGE, S. **Dynamic Languages Strike Back**. maio 2009. Disponível em: <`http://steve-yegge.blogspot.com/2008/05/dynamic-languages-strike-back.html`>. Acesso em: mar., 2009. 1

A
Implementação dos Testes de Desempenho

Neste capítulo é apresentada a implementação dos testes de desempenho apresentados no capítulo 5. Esse teste consiste de um processo servidor que publica um *servant* cujas operações são chamadas por um cliente remoto. Esse cliente mede o tempo de execução das diferentes operações oferecidas pelo *servant*.

A.1
Servidor Utilizado nos Testes

Todas as operações do *servant* são implementadas como métodos vazios, ou seja, que não realizam nenhuma operação, de forma que o tempo de cada operação reflita a sobrecarga mínima imposta pelo middleware. O servidor é escrito em Lua usando OiL e também Java e C++ usando os respectivos mapeamentos de CORBA para essas linguagens. A IDL que descreve a interface implementada pelo *servant* publicado pelo servidor utilizado no teste é apresentada na figura 5.2 na página 123.

A.2
Aplicação Cliente que Realiza os Testes

A aplicação cliente pode ser dividida em três partes principais: a infra-estrutura para coleta e gravação dos resultados; a execução dos testes com valores simples; e a execução dos testes com seqüências de valores. As seções seguintes apresentam essas partes em detalhes. Em todos os testes, é utilizado apenas uma única implementação da aplicação-cliente que é escrita em C++ e utiliza o ORB Orbacus 4.3.2.

A.2.1
Infra-estrutura para coleta e gravação dos dados de resultado

A duração das chamadas são obtidas através da operação `time` cuja implementação é apresentada abaixo. Essa função é implementa usando a operação `gettimeofday` que tem precisão de micro-segundos.

```
extern "C" {
#include <time.h>
```

```
#include <sys/time.h>
}

double time(void) {
  struct timeval v;
  gettimeofday(&v, (struct timezone *) NULL);
  /* Unix Epoch time (time since January 1, 1970 (UTC)) */
  return v.tv_sec*1.0e3 + v.tv_usec/1.0e3;
}
```

A gravação dos dados coletados é feita através da classe `Report` apresentada abaixo. Essa classe permite abrir um arquivo de texto e ir adicionando as informações obtidas num formato similar ao de tabelas Lua, de forma que essas informações possam ser facilmente manipuladas posteriormente através de um *script* Lua que gera um arquivo num formato apropriado para a geração dos gráficos.

```
#include <fstream>

class Report
{
  private:
  std::ofstream file;

  public:
  Report(const char *name) {
    char path[1024];
    sprintf(path, "<reports>", (name + g_execid).c_str());
    file.open(path, std::ios::out|std::ios::app);
    std::cout << std::endl << name << " ... ";
    std::cout.flush();
    file << g_testid << "={";
  }
  void add(int count, double time) {
    file << "[" << count << "]=" << time << ";";
  }
  void done() {
    file << "}" << std::endl;
    std::cout << "done.";
  }
};
```

A.2.2 Execução dos testes de invocação simples

A execução dos testes de invocação simples é precedida por uma etapa de aquecimento da aplicação distribuída. Nessa etapa de aquecimento, as chamadas a serem realizadas durante o teste são realizadas repetidas vezes até que a sua duração se estabilize em torno de um valor. Após essa estabilização, o teste é efetivamente realizado e os dados coletados são armazenado num arquivo com o sufixo `_calls`. Os tempos das chamadas realizadas durante a etapa de aquecimento são apresentados na figura 5.3 na página 123.

```
{ // warm-up
  Report report("<name>_calls_dist");
  int count = 0;
  for (int t = 0; t < 30; ++t) {
    double minimum = ::std::numeric_limits< double >::max();
    while (true) {
      double spent = time();
      for (int _ = 0; _ < 100; ++_) {
        sender->send_<name>(<value>);
      }
      spent = time()-spent;
      report.add(++count, spent);
      if (spent < minimum) {
        minimum = spent;
      } else {
        break;
      }
    }
  }
  report.done();
}
{ // actual tests
  for (int t = 0; t < 30; ++t) {
    Report report("<name>_calls");
    double start = time();
    for (int _ = 0; _ < 100; ++_) {
      sender->send_<name>(<value>);
    }
    report.add(g_testqual, time()-start);
    report.done();
  }
}
```

A.2.3
Execução dos testes de invocação com passagem de seqüências

A última parte da implementação dos testes consiste na execução de testes cujo parâmetro são seqüências de valores. Este teste é muito similar aos testes apresentados anteriormente, entretanto, a seqüência de valores a ser transmitida é construída antes do início do teste e é reutilizada em todas as chamadas.

```
// prelude
::ORBTests::<name>Seq block(10);
block.length(10);
for (int i = 0; i < 10; ++i) block[i] = <value>;

{ // warm-up
  Report report("<name>_seqs_dist");
  int count = 0;
  for (int t = 0; t < 30; ++t) {
    double minimum = ::std::numeric_limits< double >::max();
    while (true) {
      double spent = time();
      for (int _ = 0; _ < 100; ++_) {
        sender->send_<name>s(block);
      }
```

```
        spent = time()-spent;
        report.add(++count, spent);
        if (spent < minimum) {
          minimum = spent;
        } else {
          break;
        }
      }
    }
    report.done();
  }
  { // actual tests
    for (int t = 0; t < 30; ++t) {
      Report report("<name>_seqs");
      double start = time();
      for (int _ = 0; _ < 100; ++_) {
        sender->send_<name>s(block);
      }
      report.add(g_testqual, time()-start);
      report.done();
    }
  }
```