Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

Design Patterns in Communications Software

SIGS Reference Library

1. *Object Methodology Overview CD-ROM* • Doug Rosenberg
2. *Directory of Object Technology* • edited by Dale J. Gaumer
3. *Dictionary of Object Technology: The Definitive Desk Reference* • Donald G. Firesmith and Edward M. Eykholt
4. *Next Generation Computing: Distributed Objects for Business* • edited by Peter Fingar, Dennis Read, and Jim Stikeleather
5. *C++ Gems* • edited by Stanley B. Lippman
6. *OMT Insights: Perspectives on Modeling from the Journal of Object-Oriented Programming* • James Rumbaugh
7. *Best of Booch: Designing Strategies for Object Technology* • Grady Booch, edited by Ed Eykholt
8. *Wisdom of the Gurus: A Vision for Object Technology* • selected and edited by Charles F. Bowman
9. *Open Modeling Language (OML) Reference Manual* • Donald Firesmith, Brian Henderson-Sellers, and Ian Graham
10. *Java™ Gems: Jewels from Java™ Report* • collected and introduced by Dwight Deugo, Ph.D
11. *The Netscape Programmer's Guide: Using OLE to Build Componentware™ Apps* • Richard B. Lam
12. *Advanced Object-Oriented Analysis and Design Using UML* • James J. Odell
13. *The Patterns Handbook: Techniques, Strategies, and Applications* • edited by Linda Rising
14. *Kent Beck's Guide to Better Smalltalk: A Sorted Collection* • Kent Beck
15. *The Elements of Java Style* • Al Vermeulen et al.
16. *More Java Gems* • edited by Dwight Deugo, Ph.D.
17. *More C++ Gems* • edited by Robert C. Martin
18. *The Road to the Unified Software Development Process* • Ivar Jacobson, edited by Stefan Bylund
19. *Design Patterns in Communications Software* • edited by Linda Rising

DESIGN PATTERNS IN COMMUNICATIONS SOFTWARE

Edited by
Linda Rising

Foreword by
Douglas C. Schmidt

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

32 Avenue of the Americas, New York NY 10013-2473, USA

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9780521790406

First published in 2001

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

Design patterns in communications software / edited by Linda Rising.
p. cm. – (SIGS reference library series ; 19)
Includes bibliographical references and index.
ISBN 0-521-79040-9
1. Communications software. 2. Computer software – Development. 3. Software patterns. I. Rising, Linda. II. Series.

TK5105.9 D48 2001
005.7'13–dc21

00-065094

ISBN 978-0-521-79040-6 Hardback

To the first TelePLoP guys at ChiliPLoP. This is your baby.

Ward Cunningham
Dennis DeBruler
David DeLano
Jim Doble
Bob Hanmer
John Letourneau
Greg Stymal
Greg Utas
And Cope, our shepherd!

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

Contents

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

FOREWORD
THE CONSEQUENCES OF INFORMATION TECHNOLOGY COMMODITIZATION

INFORMATION TECHNOLOGY (IT) is rapidly becoming a commodity, where hardware and software artifacts get faster, cheaper, and better at a predictable pace. For the past decade we've benefited from the commoditization of hardware, such as CPUs and storage devices, and networking elements, such as IP routers. More recently, the maturation of programming languages, such as Java and C++, operating environments, such as POSIX and Java Virtual Machines, and middleware, such as CORBA and Enterprise Java Beans, are helping to commoditize many software layers and components, as well. Historically, however, the quality of commodity software – particularly communication software – has lagged behind hardware. Fortunately, recent improvements have enabled commodity off-the-shelf (COTS) software components to be used in an increasing number of mission-critical applications, ranging from avionics mission computing to hot rolling mills, backbone routers, and high-speed network switches.

The commoditization of IT has a number of consequences:

- *Greater focus on integration rather than programming* – There is an ongoing trend *away from* programming systems from scratch *toward* integrating them by configuring and customizing reusable frameworks and components. While it is possible in theory to program systems from scratch, economic and organizational constraints – as well as increasingly complex requirements and global competitive pressures – are making it infeasible economically to do so in practice. In the future, therefore, most systems will be configured by integrating reusable commodity components implemented by different suppliers.

- *Increased technology convergence and standardization* – To leverage inexpensive commodity hardware and software effectively, application developers are converging toward fewer general-purpose tools, platforms, and methods than they've used in the past. For example, rather than choosing from dozens of programming languages, operating systems, network protocols, and middleware, newer systems are being developed with a relatively small number of common tools and platforms, such as C++, Java, UNIX, Windows NT, TCP/IP, and CORBA, many of which are industry standards. Likewise, there's a general consolidation of development methods and modeling notations away from ad hoc technologies toward industry standards, such as UML.
- *Growing economies of scale for mass market technology and personnel* – Due to technology convergence, there is an increasing abundance of good technology and personnel available at competitive prices for mainstream mass markets, such as e-commerce or consumer electronics built using general-purpose operating systems, languages, networks, and middleware. However, for niche markets – such as real-time embedded systems written using proprietary platforms, languages, and tools – commodity hardware and software artifacts and personnel are much more expensive – if they can be found at all. Thus, the further an industry diverges from the mainstream mass market, (1) the greater the costs it incurs to develop, deploy, and maintain systems and (2) the more its customers must pay for the niche technology.
- *More opportunities for disruptive technologies and global competition* – Another consequence of IT commoditization is that industries long protected by high barriers to entry, such as the telecommunication and aerospace industries, will be much more vulnerable to disruptive technologies and global competition. For example, advances in high-performance COTS hardware and fault-tolerant middleware are making it possible to build dependable network elements ranging from PBXs to high-speed backbone routers using standard hardware and software components that cost much less than today's highly proprietary systems. In turn, the commoditization of network elements will continue to erode the profit margins that market leaders of communication equipment have enjoyed historically.

- *Lower-priced, but potentially lower-quality components* – According to John Chambers, CEO of Cisco Systems, consumers are the big winners of IT commoditization because "everything gets cheaper forever." (Clearly, John must not have purchased a house in Silicon Valley recently.) Nevertheless, global competition and time-to-market pressures are generally increasing productivity and driving commodity IT prices to marginal cost *for low- and middle-end systems.* What's not clear at this point, however, is how well commodity hardware and software can support the dependability and fidelity requirements of *high-end systems,* such as carrier class central office switches or flight-critical avionics control systems. Unfortunately, there will be little incentive to improve the quality of commodity artifacts as long as:
  - Most users emphasize price and features over quality and scalability.
  - Vendors can continue to make money selling lower-quality products inexpensively in mass quantities.
  - There's no credible competition or alternatives.
- *The decline of internally funded R&D* – Shrinking profit margins and increasing shareholder pressure to cut costs are making it hard for companies to invest heavily in long-term research that doesn't yield short-term payoffs. As a result, many companies can no longer afford the luxury of internal R&D organizations that produce proprietary hardware and software components with customized quality-of-service support. To fill this void, therefore, commodity hardware and software researched and developed by third parties is becoming increasingly strategic to many industries. This trend is requiring companies to transition away from proprietary architectures toward more open systems that can reap the benefits of externally funded R&D and open-source development processes.
- *Potential complexity cap for next-generation complex systems* – Over the past five years there has been a steady flow of faculty, staff, and grad students out of the traditional research centers, such as universities and research labs, and into startup companies and other industrial positions. While this migration has helped to fuel a global economic IT boom, it's unclear whether this trend bodes well for long-term technology innovation. In particular, without an investment in fundamental

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

> R&D it will be hard for developers of next-generation systems to master the complexities associated with the move toward large-scale distributed "systems of systems" in many domains. Thus, as the current generation of technology transitions run their course, the systemic reduction in long-term research funding relative to short-term venture capital funding may be limiting the level of complexity of systems that can be developed and integrated using commoditized hardware and software components.

In the face of the relentless drive toward IT commoditization, it has become essential for developers, companies, and even entire countries to reconsider how to remain competitive. As hardware and software components are increasingly being written, fabricated, and deployed as inexpensive commodities via low-cost digital distribution channels over the Internet, traditional market leaders and supply chains are being altered significantly. For example, the ubiquity of the Web and the maturation of operating system and middleware standards are enabling a rapid growth of open-source operating systems, such as Linux, and open-source CORBA middleware, such as MICO, omniORB, ORBacus, and TAO. Moreover, open-source business models based on industrial standards are significantly altering the pricing levels, business strategies, and licensing models for vendors of proprietary "binary-only" solutions.

How Patterns Are Helping Us Succeed in a World of Information Technology Commoditization

In a highly commoditized IT economy, human capital is an increasingly strategic asset. In the future, therefore, I believe that premium value and competitive advantage will accrue to individuals, organizations, and nations that master the patterns and pattern languages necessary to integrate COTS hardware and software to develop complex systems that can't be bought off the shelf. Patterns represent successful solutions to challenges that arise when building software in particular contexts. When related patterns are woven together, they form a language that helps to (1) define a vocabulary for talking about software development and integration challenges and (2) provide a process for the orderly resolution of these challenges.

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

Identifying, studying, and applying patterns and pattern languages can help us ensure that COTS components are created and integrated into high-quality communication systems by addressing the following system development and evolution challenges:

- *Communication of architectural knowledge among developers* – Patterns help capture essential properties of software architectures, while suppressing certain details that are not relevant at a particular level of abstraction. For example, patterns help to document software architectures by expressing the structure and dynamics of participants at a level higher than (1) source code or (2) object-oriented (OO) design models that focus on individual objects and classes. Patterns also provide software organizations with shared vocabularies and common architectural models, thereby helping to improve communication within and across developers, teams, and projects.

- *Accommodating new design paradigms or architectural styles* – Historically, the adoption of COTS and other technology advances has been limited by the effort required to transition developers trained in traditional techniques to newer paradigms, such as OO analysis, design, and programming. Fortunately, many core patterns have originated in non-OO contexts, such as operating system kernels, I/O subsystems, and databases, that are familiar to developers of legacy systems. Thus, developers who study these patterns can more readily migrate to new paradigms by learning how to leverage and extend domain expertise they gained from their prior experience and express it effectively in newer languages, platforms, methods, and notations.

- *Resolving key nonfunctional forces associated with accidental complexity* – A surprising amount of software complexity is accidental and arises from limitations with development tools and platforms. For instance, the vestiges of legacy hardware and software platform diversity have made it hard to develop portable applications and tools that can run across multiple operating environments, such as POSIX, Win32, and embedded real-time operating systems. These accidental complexities are being addressed today by identifying key patterns and pattern languages and reifying them into reusable middleware, such as CORBA or real-time Java, that shields developers from many tedious, error-prone, and nonportable programming details.

- *Avoiding development traps and pitfalls that were historically learned by costly trial and error* – Developers of communication systems must address many recurring design challenges related to key properties, such as efficiency, scalability, robustness, and extensibility. For decades, successful architects and developers have learned to resolve these challenges by trial and error during a lengthy apprenticeship process. At any point in time, however, the knowledge gained from this process resides largely in the heads of domain experts or buried deep within complex system source code. By externalizing this expertise in the form of patterns and pattern languages, new generations of developers can learn how to avoid common traps and pitfalls in their domains without requiring such an extensive apprenticeship.

Manifesting the Maturation of the Patterns Community

Over the past seven years an extensive body of literature on patterns and pattern languages has emerged that identifies, documents, and catalogs successful families of solutions to common software challenges. This literature has improved the construction of commercial software significantly by enabling the widespread reuse of software architectures, developer expertise, and OO application framework components. As a consequence, we now have a good collective understanding of how to design and implement certain types of software components using off-the-shelf tools and methods.

Much of the patterns literature has focused on a few well-traveled domains, however, such as graphical user-interface frameworks or business applications. As the scope and criticality of communication system requirements continue to expand, our most pressing need has become elevating the focus from relatively small-scale, stand-alone software mechanisms, protocols, and patterns to large-scale, distributed policies, architectures, and pattern languages. Developers of communication systems face many vexing inherent complexities, such as partial failures, distributed deadlock, and end-to-end QoS enforcement, that contain many interlocking aspects. Therefore, unless the panoply of commodity hardware and software point solutions can be consolidated into integrated frameworks, their value will be diminished and can in fact make matters worse instead of better.

Until now, no single book has focused on patterns and pattern languages for communication systems with mission-critical requirements, such as telecommunications systems with 24×7 high-availability requirements or real-time middleware that can provide stringent end-to-end latency and jitter guarantees to applications. The material in this book exemplifies the maturation of the patterns community and is a major contribution to the study of patterns and pattern languages for communication systems. By learning this material you'll be able to better design and implement communication systems that can't be bought off the shelf, thereby improving the productivity and quality of your software solutions and staying ahead of your competition.

We are fortunate that Linda Rising has found time in her busy life to assemble the work of an outstanding group of authors for this book. During the past ten years I've had the honor to meet and work with most of these authors, many of whom are internationally renowned experts in their fields. If you want thorough coverage of the patterns and pattern languages that are shaping next-generation communication software, read this book. I've learned much from it and I'm confident that you will too.

Douglas C. Schmidt
University of California, Irvine

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

ACKNOWLEDGMENTS

I WAS THE CONFERENCE CHAIR for the first ChiliPLoP in 1998 and was not able to attend the TelePLoP hot topic. This publication was conceived at that gathering, but it was only when the time suddenly became available that I was able to implement it.

I have tried to capture the spirit of the participants' intent even though I'm somewhat removed in time and distance!

Thanks to all the TelePLoPers and all the other contributors of communications patterns. These people are the inspiration for this book and the creators of its content. All I have done is to line 'em up and move 'em out!

Thanks to the good folks at Cambridge University Press, especially Lothlórien Homet, who believed that this book was important and should be published.

Thanks to my husband, Karl Rehmer, for being happy and proud about what I do and supporting and encouraging me along the way.

Linda Rising
Denmark 2001

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

Contributors

Michael Adams is a telecommunications software engineer with 11 years of experience with AT&T, Lucent Technologies, and Idea Integration. He currently runs Timberway International (www.timberway.com), a company specializing in helping individuals and small businesses market their products on the Internet. He likes to spend his off hours practicing and teaching Wing Tsun Kung Fu. madams@timberway.com

Chris Cleeland graduated with a B.S. in Computer Engineering from Tulane University and currently works for Object Computing, Inc. in St. Louis, Missouri, where he develops and teaches industry courses in CORBA and Design Patterns, consults in the realm of networking and Distributed Computing, and helps support The ACE ORB, an open-source ORB from the Distributed Object Computing (DOC) Center at Washington University, St. Louis, of which he was a founding researcher. His software development experience spans two decades and several areas such as health care, manufacturing automation, and telecommunications. Chris thanks his parents for providing him with a solid foundation, his wife Pat and family for their constant loving support, and the late W. Richard (Rich) Stevens, a close friend whose writings continue to inspire aspirations of greatness. cleeland_c@ociweb.com

James O. Coplien is a member of the Software Production Research Department in Bell Laboratories of Lucent Technologies. He holds a B.S. in Electrical and Computer Engineering and an M.S. in Computer Science, both from the University of Wisconsin at Madison, and a Ph.D. from Vrije Universiteit Brussel. He is currently studying organization communication patterns to help guide process evolution. His other research areas include multi-paradigm design and architectural patterns of telecommunication software. He is author of *C++ Programming Styles and Idioms,* the foremost high-end C++ book in the industry, and of *Multi-Paradigm Design for C++*. He was co-editor of two volumes of *Pattern Languages of Program Design,* and he writes a patterns column for the *C++ Report.* He is a Member Emeritus of the Hillside Group. He was program chair of ACM OOPSLA '96 and program co-chair of the First International South Pacific Conference on Pattern Languages of Program Design. cope@research.bell-labs.com

Fernando Das Neves is a Ph.D. candidate at Virginia Tech. He was formerly at Universidad Nacional de La Plata, Argentina, where the *Bodyguard* pattern (included in this publication) was originally written. His research interests include information retrieval and visualization and object-oriented programming. fdasneve@vt.edu

Dennis L. DeBruler is a Distinguished Member of Technical Staff at Bell Laboratories of Lucent Technologies. He received a B.S.E.E. from Northwestern and an M.S. in Computer Science from Purdue University. His work at Bell Laboratories during the 1970s concerned software development tools for the custom processors used in electronic switching systems. In the 1980s and 1990s he helped architect the software and hardware of various telecommunication "boxes" (packet switch, ATM terminal adapter, video server, speech recognition). He has also helped design, implement, and test the software for most of these boxes. Dennis attended the first PLoP conference, for which he submitted a paper, and he has shepherded at least one paper for nearly every PLoP since then. His pattern language on real-time scheduling was workshopped at one of the TelePLoP tracks at ChiliPLoP. debruler@lucent.com

Michael Duell is a software designer at AG Communication Systems in Phoenix, Arizona. He is currently working on the Communications Assistance for Law Enforcement Act feature. He has published papers on software design patterns, including: "Non-Software Examples of Software Design Patterns" in *Object* magazine, July 1997; "Managing Change through Patterns" in *IEEE Communications* magazine, April 1999; and "Non-Software Examples of Patterns of Software Architecture" in *JOOP,* May 2000. He has presented papers at OOPSLA '96 and IEEE COMPSAC '99. He has also organized workshops and poster sessions on non-software pattern examples at OOPSLA '97 and '98. He holds a B.S. in Computer Science and Mathematics from the University of California, Davis, an M.S. in Telecommunications from Southern Methodist University and an M.B.A. from Arizona State University. duellm@agcs.com; http://www.agcs.com/supportv2/techpapers/patterns/papers/tutnotes/index.htm

Robert Gamoke is a Distinguished Member of Technical Staff at Lucent

Technologies in Naperville, Illinois. After joining Western Electric in 1980, he has served as the principal software architect for the 1A and 1B processor used in the 4ESS switch. He continues to develop features and to mine patterns from the designs of that switch. gamoke@lucent.com

Alejandra Garrido has been a graduate student at the University of Illinois at Urbana-Champaign since 1997. She was formerly at Universidad Nacional de La Plata, Argentina, as a member of the research group Lifia. Her research interests include object-oriented frameworks, design patterns, and software refactoring. She obtained an M.S. in Computer Science from UIUC in May 2000. Her M.S. thesis and her current research are in refactoring C programs. garrido@students.uiuc.edu

Christopher D. Gill is a Research Associate in the Computer Science Department at Washington University in St. Louis, Missouri, working in the Center for Distributed Object Computing. He conducts research on quality of service (QoS) patterns and frameworks for distributed object computing. He is a doctoral candidate in the Computer Science Department at Washington University, and is planning to graduate in May 2001. While at the Center for Distributed Object Computing, he has made research contributions in the areas of hybrid static/dynamic processor scheduling strategies, adaptive resource management, and empirical measurement and visualization of dynamic real-time performance. cdgill@cs.wustl.edu http://www.cs.wustl.edu/~cdgill

Robert Hanmer is a Consulting Member of Technical Staff at Lucent Technologies. He is a past Program Chair for the Pattern Languages of Programming Conference held at Allerton Park. He is active in the TelePLoP group, an informal collection of pattern advocates and authors interested in the field of telecommunications. He is a co-author of a collection of patterns in the PLoPD2 book on reliable system design, which has been republished in *The Patterns Handbook: Techniques, Strategies, and Applications*. The Telecommunications Input Output pattern language that he co-authored was published in the PLoPD4 book. Within Lucent, he is active as an advocate and teacher of patterns and designing for reliability. He has been learning the patterns of the 4ESS switch since 1987. He holds B.S. and M.S. degrees in Computer Science from Northwestern University. hanmer@lucent.com

Neil B. Harrison is a Distinguished Member of Technical Staff at Avaya, Inc. He works in telecommunications software, software tools, and software process and organization studies. He was an early advocate of patterns and regularly teaches pattern courses. He has published patterns and articles on patterns in numerous books and was the lead editor of *Pattern Languages of Program Design,* volume 4. His leadership in pattern "shepherding" has been recognized through the "Neil Harrison Shepherding Award," which is awarded at PLoP conferences. He is also a Past Program Chair of ChiliPLoP. Together with Jim Coplien, he has conducted extensive studies on software organizations and has published articles in this field. He received a B.S. from Brigham Young University, with high honors and university scholar designations, and an M.S. from Purdue University, both in computer science. nbharrison@avaya.com

Fred Keeve recently retired from Lucent Technologies after a career of telecomunication system design and development.

Fred Kuhns is a Senior Research Associate in the Applied Research Laboratory at Washington University. He received the M.S.E.E. from Washington University, St. Louis, and the B.S.E.E. from the University of Memphis, Tennessee. His research interests include operating system and network support for high-performance, real-time distributed object computing systems. His recent research projects have focused on the design and implementation of real-time middleware, I/O subsystems, high-performance interfaces, and integrated service routers. His current projects include developing multiservice routers with integrated support for active networking and different QoS models. fredk@arl.wustl.edu; http://www.arl.wustl.edu:/~fredk

David L. Levine is Director of the Center for Distributed Object Computing of the Department of Computer Science, Washington University in St. Louis. Dr. Levine's current research interests include development of efficient object-oriented middleware for embedded systems and testing and performance analysis of real-time systems. In addition, he helps maintain the Adaptive Communications Environment (ACE) framework and The ACE ORB (TAO). levine@cs.wustl.edu; http://www.cs.wustl.edu/~levine

Gerard Meszaros is Chief Architect at Clearstream Consulting, consulting resources in applying advanced software development techniques, such as eXtreme programming to help Clearstream's clients achieve faster and higher quality systems solutions. He helps clients apply business modeling and object and component technology to a variety of problem domains including telecommunication, e-commerce, and gas transportation. Previously, he served as Chief Architect on several major projects at Nortel Networks' R&D

subsidiary, Bell Northern Research. He has had patterns published in each of the first three PLoPD books and has been invited to serve on panels, conduct workshops, and present papers and tutorials at OOPSLA and other conferences. He uses patterns actively in designing software and in mentoring software developers, architects and project managers in good software design practices. He holds a B.Sc. (Honors) in Computer Science from University of Manitoba. gerard.meszaros@acm.org; http://www.clearstreamconsulting.com

Keith Nicodemus received an A.T.&T. Bell Laboratories Fellow award for 1987. He retired from A.T.&T. (now Lucent) in November 1995, and since then he has worked intermittently for Lucent at Bell Laboratories. He holds a B.S.E.E. from the University of Iowa and an M.S.E.E. from Stevens Institute of Technology. knicodemus@lucent.com

Carlos O'Ryan is a Research Assistant of the Distributed Object Computing Laboratory and graduate student of the Department of Computer Engineering, University of California, Irvine. In that position he participates in the development of TAO (The ACE ORB), an Open Source, real-time, high-performance, CORBA-compliant ORB. Before his current position he served as senior developer in GBO, a Chilean company that develops distributed interactive simulation systems for the army and air force of that country. He also participated in the deployment of this system for the armies of Mexico and El Salvador. He obtained his B.S. in Mathematics from the Pontificia Universidad Catolica de Chile in 1992 and an M.S. in Computer Science from Washington University in 1999. coryan@cs.wustl.edu; http://doc.ece.uci.edu/~coryan/

Don S. Olson is a Senior Engineer with AG Communication Systems. With a B.S. in Mathematics from the University of Alabama in Huntsville, Alabama, he has worked in software development for more than 23 years in industries as diverse as CAD/CAM systems, space exploration, commercial flight systems, and telecommunications. He is a Strowger Award recipient for his contributions to a scalable broker-based architecture, and he has received numerous software design awards for his work in automatic call distribution systems, simulators, and real-time database protocols. He is a former Program Chair for ChiliPLoP in Wickenburg, Arizona, contributor to *The Patterns Handbook,* and presented to the plenary session of the Intelligent Networks workshop in Colorado Springs, Colorado, in 1997. He is a frequent visitor and participant in discussions on WikiWikiWeb, a patterns forum, and is currently working on a book about alternative management for software development. olsond@agcs.com

Cambridge University Press
978-0-521-79040-6 - Design Patterns in Communications Software
Edited by Linda Rising
Frontmatter
More information

Contributors

Ossama Othman is a Research Assistant at the Distributed Object Computing Laboratory in the Department of Electrical and Computer Engineering, University of California at Irvine. He is currently pursuing his Ph. D. studies and research in secure, scalable, and high availability CORBA-based middleware. As part of that work, he is one of the core development team members for the open source CORBA ORB TAO (The ACE ORB). Prior to his work in CORBA-based middleware, he served as a computer-imaging specialist at the Center for Radiophysics and Space Research at Cornell University, where he developed software for use in the analysis of NASA Galileo spacecraft images, such as those of the Shoemaker-Levy comet crash on Jupiter. ossama@uci.edu; http://www.ece.uci.edu/~ossama/

Jeff Parsons received his B.S. degree in Computer Science from Washington University in St. Louis, Missouri. He is currently employed as a Research Associate at the Center for Distributed Object Computing, also at Washington University, as a member of the TAO development team, and is pursuing an M.S. degree. His professional interests include object-oriented design patterns and frameworks, distributed object middleware, and CORBA. parsons@cs.wustl.edu; http://siesta.cs.wustl.edu/~parsons/

Nat Pryce is a Senior Software Engineer at Quokka Sports in San Francisco and London. He is part of a team building distributed systems that deliver real-time visualization of sports events over the Web. Previously he worked as a research associate at Imperial College, London, where he obtained a Ph.D. for his research into architectures for component-based, distributed systems.Nat.Pryce@quokka.com

Linda Rising has a Ph.D. from Arizona State University in the area of object-based design metrics. Her background includes university teaching experience as well as work in industry in the areas of telecommunications, avionics, and strategic weapons systems. She has presented a number of tutorials and workshops at OOPSLA and other conferences. She was the Conference Chair for the first Southwestern patterns conference, ChiliPLoP. She is the editor of *A Patterns Handbook.* She was the editor of a special issue of *IEEE Communications* on Design Patterns in Communications Software and the editor of *A Pattern Almanac 2000.* risingl@acm.org

Douglas C. Schmidt is an Associate Professor in the Electrical and Computer Engineering Department at the University of California, Irvine. He is currently serving as a Program Manager at the DARPA Information Technology Office where he leads the national effort on distributed object computing middleware research. His research focuses on design patterns, optimization principles, and empirical analyses of object-oriented techniques that facilitate the development of high-performance, real-time distributed object computing middleware on parallel processing platforms running over high-speed networks and embedded system interconnects. He is an internationally recognized and widely cited expert on distributed object computing patterns, middleware frameworks, and real-time CORBA, and he has published widely in top IEEE, ACM, IFIP, and USENIX technical journals, conferences, and books. His publications cover a range of experimental systems topics including high-performance communication software systems, parallel processing for high-speed networking protocols, real-time distributed object computing with CORBA, and object-oriented design patterns for concurrent and distributed systems. schmidt@uci.edu; http://www.ece.uci.edu./~schmidt/

Greg Stymfal obtained a B.S. in Electrical Engineering from the University of Colorado. Since that time he has developed, tested, and managed software in the embedded telecommunications industry. He has been involved with software and organizational patterns since 1994. The Telecommunications Input Output pattern language that he co-authored was published in the PLoPD4 book. He is currently a Product Manager for VoIP solutions at AG Communication Systems. stymfalg@agcs.com

Junichi Suzuki received B.S., M.S., and Ph.D. degrees in Computer Science from Keio University, Japan. He is currently a research fellow at the Department of Information and Computer Science at University of California, Irvine. Before his current position, he worked for the Object Management Group Japan, Inc. as a technical director. His research interests include object-oriented patterns and frameworks, computational reflection, object-oriented development methodology, distributed object computing, intelligent user interface, agent communication, and computational biology. He is a member of ACM, IEEE-CS, IPSJ, and JSSST. suzuki@yy.cs.keio.ac.jp

Greg Utas obtained an Honors B.Sc. in Computer Science from the University of Western Ontario in 1979. Since joining Nortel Networks in 1981, he has served as the principal architect of the call processing frameworks for DMS-100 Centrex services, the DMS-100 International switch, and Nortel's GSM Mobile Switching Center. He has presented papers at the International Switching Symposium, the International Workshop on Feature Interactions in Telecommunications and Software Systems, and at the TelePLoP track at ChiliPLoP. He is currently Chief Software Architect, Mobility Services Development. utas@nortelnetworks.com

Just A. van den Broecke of Just Objects B.V. (http://www.justobjects.nl) is a freelance software architect/developer/trainer specializing in distributed object technology. Currently he spends most of his time using Java and Web technologies like XML in projects that range from e-commerce banking applications to multiuser-multimedia tools for performing artists (http://www.keyworx.org). At the time of writing the article in this publication, he was working at the Forward Looking Work department within Lucent Technologies in Huizen, The Netherlands. just@justobjects.nl

Yoshikazu Yamamoto received the B.S., M.S., and Ph.D. degrees in Administration Engineering from Keio University, Tokyo, Japan. He is currently an Associate Professor in the Department of Computer and Information Science at Keio University. He worked at Linkoping University, Sweden, as a visiting professor from 1981 to 1983. His current research interests include distributed discrete event simulation and modeling, OOP, agent programming, intelligent interface, and documentation. He is a member of ACM, IEEE-CS, IPSJ, and the director board of JSSST. yama@ics.keio.ac.jp; http://www.yy.ics.keio.ac.jp/