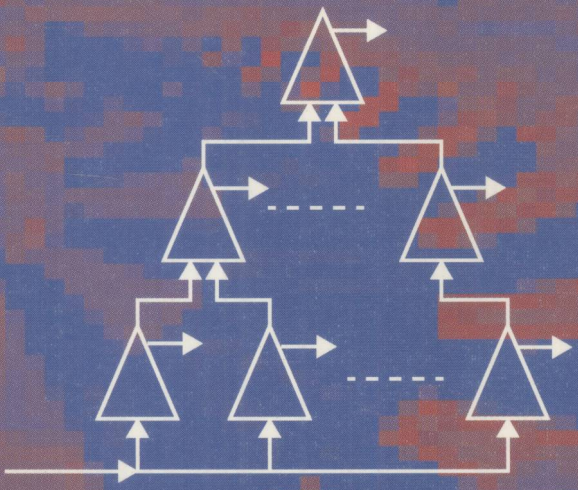
State-of-the-Art Survey

LNCS 3069

Rogério de Lemos
Cristina Gacek
Alexander Romanovsky (Eds.)

Architecting Dependable Systems II

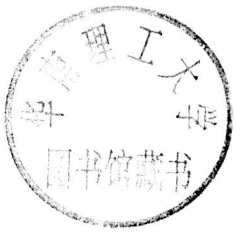


Springer

TP303
A673

Rogério de Lemos
Cristina Gacek
Alexander Romanovsky (Eds.)

Architecting Dependable Systems II



Springer

Volume Editors

Rogério de Lemos
University of Kent, Computing Laboratory
Canterbury, Kent CT2 7NF, UK
E-mail: r.delemos@kent.ac.uk

Cristina Gacek
Alexander Romanovsky
University of Newcastle upon Tyne, School of Computing Science
Newcastle upon Tyne, NE1 7RU, UK
E-mail: {Cristina.Gacek, Alexander.Romanovsky}@ncl.ac.uk

Library of Congress Control Number: 2004113649

CR Subject Classification (1998): D.2, D.4

ISSN 0302-9743
ISBN 3-540-23168-4 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11314646 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3069

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Foreword

Enforcing the dependability of software systems has been a very active and productive area of research for over 30 years, addressing support for fault prevention, fault tolerance, fault removal and fault forecasting. Such an effort has in particular led to introducing a number of dependability concepts, and related principled methods and tools guiding the structuring of dependable systems, and further allowing reasoning about the systems' dependability. As such, research results in the dependability area impact upon the overall software development process. However, dependability has for long been considered as an aside property in the development of software systems, except for specific classes of systems such as safety-critical systems that cannot tolerate unexpected behavior following the occurrence of failures.

The increasing reliance on software systems that are now surrounding our everyday's life, being embedded in most devices, and providing us with access to a huge amount of content and services via the Internet, makes dependability a prime requirement for today's systems. In particular, dependability requirements should be accounted for in the early phase of the development process, since dependability means significantly impact design choices. In this context, architectural modelling of software systems offers much benefit towards assisting the design of dependable systems and assessing their dependability. By abstracting the low-level details of the system, architectural modelling allows effective exploitation of formal methods for reasoning about the systems' behavior, which constitutes a key dependability means. Architectural modelling further allows developers to comprehensively deal with dependability requirements in the structuring of their systems.

Bringing together researchers from the dependability and software architecture communities, via dedicated workshops and books on improvement to the state of the art on architecting dependable systems, can only be acknowledged as a valuable effort towards eliciting architecture modelling languages, methods and tools that support the thorough development of dependable software systems. This book, the second of an undoubtedly promising series, introduces research results that show how dependability and software architecture research conveniently complement and benefit from each other, addressing specific system architecting for dependability, integration of fault tolerance means with software architecture, and architecture modelling for dependability analysis. Last but not least, reports on industrial experience highlight how such an effort meets industrial practices in dependable software system development. As a result, the breadth and depth of the coverage that is provided by this book on recent research in architecting dependable systems is particularly impressive, and the editors and authors are to be congratulated.

June 2004

Valérie Issarny
INRIA
Research Unit of Rocquencourt

此为试读,需要完整PDF请访问: www.ertongbook.com

Preface

System dependability is defined as reliance that can be justifiably placed on the service delivered by the system. It has become an essential aspect of computer systems as everyday life increasingly depends on software. It is therefore a matter for concern that dependability issues are usually left until too late in the process of system development. This is why, even though there is a large body of research on dependability, reasoning about dependability at the architectural level is only just emerging as an important theme. It is a theme that needs to be actively pursued since architectural representations have been shown to be effective in helping to understand broader system characteristics by abstracting away from details. Apart from this, there are other factors that make it urgent to consider dependability at the architectural level, such as the complexity of emerging applications and the need for building trustworthy systems from the existing untrustworthy components.

This book comes as a result of an effort to bring together the research communities of software architectures and dependability. It was inspired by the ICSE 2003 Workshop on Software Architectures for Dependable Systems (WADS 2003), where many interesting papers were presented and lively discussions took place. The book addresses issues that are currently relevant to improving the state of the art in architecting dependable systems. It presents a selection of peer-reviewed papers stemming from some original WADS 2003 contributions and several invited ones. The book consists of four parts: architectures for dependability, fault tolerance in software architectures, dependability analysis in software architecture, and industrial experience.

The first part of this book focuses on software architectures for dependability. Its first paper, by Koutsoukos, Loureno, Avillez, Gouveia, Andrade, Fiadeiro, and Wermelinger, is entitled "Enhancing Dependability Through Flexible Adaptation to Changing Requirements". This paper describes an architectural approach that relies on coordination contracts to facilitate the dynamic adaptation of systems to changing domain rules. The approach is illustrated through a case study in the financial systems area, where agreed policies and conditions are negotiated on a case-by-case basis. The paper concludes by reporting on an information system that ATX Software developed for a company specialized in recovering bad credit. The second paper in this part is "A Self-optimizing Run-Time Architecture for Configurable Dependability of Services" by Tichy and Giese. In this paper, Tichy and Giese identify a set of architectural principles that can be used to improve the dependability of service-based architectures. These architectural principles have been instantiated by extending Jini, and have been evaluated qualitatively and quantitatively for a configuration of multiple identical services, showing how the different parameters affect the resulting dependability. The paper by Knight and Strunk on "Achieving Critical System Survivability Through Software Architectures" addresses the idea of making a system survivable rather than highly reliable or highly available by exploring the motivation for survivability, how it might be used, what the concept means in a precise and testable sense, and how it is being implemented in two very different application areas. The subsequent

paper is authored by Rodrigues, Roberts and Emmerich. It is on "Reliability Support for the Model Driven Architecture" and elaborates on how the provision of reliability can be suitably realized through Model Driven Architectures (MDA). It is based on a platform-independent reference model that can be mapped to specific platforms. The UML metamodeling language is extended to show how design profile elements reflect on the deployment of the components when transformation rules are applied to the model. The last paper in this part, "Supporting Dependable Distributed Applications Through a Component-Oriented Middleware-Based Group Service" by Saikoski and Coulson, presents a group-based middleware platform that aims at supporting flexibility for controlled redundancy, replication, and recovery of components and services. This flexibility is provided at design time, deployment time and run-time. Their approach is based on concepts of software component technology and computational reflection.

The second part of this book is related to fault tolerance in software architectures. In the first paper, "Architecting Distributed Control Applications Based on (Re-) Configurable Middleware", Deconinck, De Florio and Belmans introduce the DepAuDE architecture developed for industrial distributed automation applications. This architecture provides a fault tolerance middleware, a library for error detection and recovery and fault treatments, and a specialized language called ARIEL for specifying fault tolerance and configuration actions. The paper concludes with a thorough discussion of a case study: a demonstrator of a Primary Substation Automation System controlling a substation for electricity distribution. The second paper of this part is entitled "A Dependable Architecture for COTS-Based Software Systems Using Protective Wrappers". The authors, Guerra, Rubira, Romanovsky and de Lemos, combine the concepts of an idealized architectural component and protective wrappers to develop an architectural solution that provides an effective and systematic way for building dependable software systems from COTS software components. The approach is evaluated using a PID controller case study. The next paper entitled "A Framework for Reconfiguration-Based Fault-Tolerance in Distributed Systems" is co-authored by Porcarelli, Castaldi, Di Giandomenico, Bondavalli and Inverardi. In this framework fault tolerance of components-based applications is provided by detecting failures using system monitoring, and by recovery employing system reconfiguration. The framework is based on Lira, an agent distributed infrastructure employed for component and application level monitoring and reconfiguration, and a decision maker used for selecting new configurations using the feedbacks provided by the evaluation of stochastic Petri net models. In the next paper, "On Designing Dependable Services with Diverse Off-The- Shelf SQL Servers", Gashi, Popov, Stankovic and Strigini argue, based on empirical results from their ongoing research with diverse SQL servers, in favor of diverse redundancy as a way of improving dependability and performance of a SQL server. The paper provides evidence that current data replication solutions are insufficient to protect against the range of faults documented for database servers, outlines possible fault-tolerant architectures using diverse servers, discusses the design problems involved, and offers evidence of performance improvement through diverse redundancy. The last paper of part two, "A New Model and a Design Approach to Building Quality of Service (QoS) Adaptive Systems", is co-authored by Ezhilchelvan and Shrivastava. The focus is on developing Internet-based services provisioning systems. The authors propose a system architecture and identify

a model appropriate for developing distributed programs that would implement such systems. The probabilistic asynchronous model proposed abstracts the network performance and dependability guarantees typically offered by the Internet service providers. The system architecture prescribes the role of QoS management algorithms to be: evaluating the feasibility of QoS requests from the end users and adapting system protocols in response to changes in the environment.

Part three of this book deals with dependability analysis in software architectures. In the first paper, which is entitled "Multi-view Software Component Modeling for Dependability", the authors Roshandel and Medvidovic focus on a more comprehensive approach for modelling components. Instead of relying just on the description of the components interfaces, and their respective pre- and post-conditions, the authors propose an approach to modelling components using four primary functional aspects of a software component (known as the Quartet): interface, static behavior, dynamic behavior, and interaction protocol. In addition to describing individually the four aspects, the paper also discusses the relationships between them for ensuring their compatibility. The goal of the work is to obtain support for the architectural-level modelling and analysis of system dependability, in particular, reliability. The second paper, "Quantifiable Software Architecture for Dependable Systems of Systems" by Liang, Puett and Luqi presents an approach for the development and evolution of dependable systems-of-systems. Based on the architectural description of these systems, the approach involves establishing consensus between the different dependability attributes associated with component systems, and translating them into quantifiable constraints. The approach illustrates that with reusable architectural facilities and associated tools support, the quantifiable architecture with multiple perspectives can be effective in supporting the engineering of dependable systems-of-systems. In the last paper of this part, which is entitled "Dependability Modeling of Self-healing Client-Server Applications", the authors Das and Woodside present an analytical model for evaluating the combined performance and dependability attributes of fault-tolerant distributed applications. The authors consider a layered software architecture in which the application and management components can fail and be repaired. It also considers the management of connections, and the application's layered failure dependencies, together with the application performance. In order to show the capability of the approach in evaluating large-scale systems, the authors apply their analytical model to an air traffic control system.

The final part of the book contains two papers that report on existing industrial experiences involving dependability in the context of software architectures. In the first paper, entitled "A Dependable Platform for Industrial Robots", the authors Mustapic, Andersson, Norström and Wall discuss the design of an open software platform for an ABB Robotic System. For them a software platform is the basis for a product-line architecture that aims to increase the number of variations between the different software systems, while maintaining the integrity of the whole robotic system. An initial step in their approach is to model at the architectural level the quality constraints of the platform, which include several dependability attributes. The second paper of this final part, "Model Driven Architecture an Industry Perspective" by Raistrick and Bloomfield, discusses some of the research work that is currently being undertaken within the avionics industry on the usage of Model Driven Architectures (MDA), an initiative of the Object

Management Group (OMG). It has been recognized that the MDA approach might become fundamental in reducing costs in the development and maintenance of software. The authors of this paper identify several fronts in which the usage of the MDA might be effective. These are: the automation of software development with the support of tools, the management of legacy systems, the mapping of avionic applications into standard modular computer systems, and the incremental certification of avionics systems.

We believe that the introduction of the topic of architecting dependable systems is very timely and that work should continue in this area. The first book of the same title, published in the summer of 2003, included expanded papers based on selected contributions to the WADS ICSE 2002 workshop and a number of invited papers. The forthcoming ICSE/DSN 2004 Twin Workshops on Architecting Dependable Systems is another ambitious project, which aims to promote cross-fertilization between the communities of software architectures and dependability.

As editors of this book, we are certain that its contents will prove valuable for researchers in the area and are genuinely grateful to the many people who made it possible. Our thanks go to the authors of the contributions for their excellent work, the WADS 2003 participants for their active support and lively discussions, and Alfred Hofmann from Springer-Verlag for believing in the idea of this book and helping us to get it published. Last but not least, we appreciate the time and effort our reviewers devoted to guaranteeing the high quality of the contributions. They are D. Akehurst, T. Bloomfield, A. Bondavalli, F.V. Brasileiro, M. Castaldi, G. Deconinck, F. Di Giandomenico, M. Correia, G. Coulson, I. Crnkovic, S. Crook-Dawkins, W. Emmerich, J.L. Fiadeiro, G. Fohler, P. Inverardi, V. Issarny, J. Knight, N. Levy, N. Medvidovic, C. Norstrom, A. Pataricza, P. Popov, S. Riddle, G. Roberts, C.M.F. Rubira, S. Shrivastava, F. van der Linden, P. Veríssimo, M. Wermelinger, C.M. Woodside, and several anonymous reviewers.

June 2004

Rogério de Lemos
Cristina Gacek
Alexander Romanovsky

Lecture Notes in Computer Science

For information about Vols. 1–3180

please contact your bookseller or Springer

Vol. 3293: C.-H. Chi, M. van Steen, C. Wills (Eds.), Web Content Caching and Distribution. IX, 283 pages. 2004.

Vol. 3274: R. Guerraoui (Ed.), Distributed Computing. XIII, 465 pages. 2004.

Vol. 3273: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), <<UML>> 2004 - The Unified Modelling Language. XIII, 454 pages. 2004.

Vol. 3271: J. Vicente, D. Hutchison (Eds.), Management of Multimedia Networks and Services. XIII, 335 pages. 2004.

Vol. 3270: M. Jeckle, R. Kowalczyk, P. Braun (Eds.), Grid Services Engineering and Management. X, 165 pages. 2004.

Vol. 3269: J. López, S. Qing, E. Okamoto (Eds.), Information and Communications Security. XI, 564 pages. 2004.

Vol. 3266: J. Solé-Pareta, M. Smirnov, P.V. Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, R.J. Gibbens (Eds.), Quality of Service in the Emerging Networking Panorama. XVI, 390 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004. (Subseries LNAI).

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004. (Subseries LNAI).

Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), Object-Oriented and Internet-Based Technologies. XII, 239 pages. 2004.

Vol. 3262: M.M. Freire, P. Chemouil, P. Lorenz, A. Gravey (Eds.), Universal Multiservice Networks. XIII, 556 pages. 2004.

Vol. 3261: T. Yakhno (Ed.), Advances in Information Systems. XIV, 617 pages. 2004.

Vol. 3260: I.G.M.M. Niemegeers, S.H. de Groot (Eds.), Personal Wireless Communications. XIV, 478 pages. 2004.

Vol. 3258: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004. XVII, 822 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004. (Subseries LNAI).

Vol. 3256: H. Ehrig, G. Engels, F. Parisi-Presicce, G. Rozenberg (Eds.), Graph Transformations. XII, 451 pages. 2004.

Vol. 3255: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Advances in Databases and Information Systems. XI, 423 pages. 2004.

Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. XVI, 910 pages. 2004.

Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. X, 397 pages. 2004.

Vol. 3250: L.-J. (LJ) Zhang, M. Jeckle (Eds.), Web Services. X, 301 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004. (Subseries LNAI).

Vol. 3246: A. Apostolico, M. Melucci (Eds.), String Processing and Information Retrieval. XIV, 332 pages. 2004.

Vol. 3245: E. Suzuki, S. Arikawa (Eds.), Discovery Science. XIV, 430 pages. 2004. (Subseries LNAI).

Vol. 3244: S. Ben-David, J. Case, A. Maruoka (Eds.), Algorithmic Learning Theory. XIV, 505 pages. 2004. (Subseries LNAI).

Vol. 3243: S. Leonardi (Ed.), Algorithms and Models for the Web-Graph. VIII, 189 pages. 2004.

Vol. 3242: X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII. XX, 1185 pages. 2004.

Vol. 3241: D. Kranzlmüller, P. Kacsuk, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XIII, 452 pages. 2004.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004. (Subseries LNBI).

Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 444 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004. (Subseries LNAI).

Vol. 3236: M. Núñez, Z. Maamar, F.L. Pelayo, K. Pousttchi, F. Rubio (Eds.), Applying Formal Methods: Testing, Performance, and M/E-Commerce. XI, 381 pages. 2004.

Vol. 3235: D. de Frutos-Escrig, M. Nunez (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2004. X, 377 pages. 2004.

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3231: H.-A. Jacobsen (Ed.), Middleware 2004. XV, 514 pages. 2004.

Vol. 3230: J.L. Vicedo, P. Martínez-Barco, R. Muñoz, M.S. Noeda (Eds.), Advances in Natural Language Processing. XII, 488 pages. 2004. (Subseries LNAI).

Vol. 3229: J.J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence. XIV, 744 pages. 2004. (Subseries LNAI).

Vol. 3226: M. Bouzeghoub, C. Goble, V. Kashyap, S. Spaccapietra (Eds.), Semantics for Grid Databases. XIII, 326 pages. 2004.

Vol. 3225: K. Zhang, Y. Zheng (Eds.), Information Security. XII, 442 pages. 2004.

Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), Recent Advances in Intrusion Detection. XII, 315 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logics. VIII, 337 pages. 2004.

Vol. 3222: H. Jin, G.R. Gao, Z. Xu, H. Chen (Eds.), Network and Parallel Computing. XX, 694 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3219: M. Heisel, P. Liggesmeyer, S. Wittmann (Eds.), Computer Safety, Reliability, and Security. XI, 339 pages. 2004.

Vol. 3217: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 1114 pages. 2004.

Vol. 3216: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 930 pages. 2004.

Vol. 3215: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVII, 906 pages. 2004. (Subseries LNAI).

Vol. 3214: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1302 pages. 2004. (Subseries LNAI).

Vol. 3213: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1280 pages. 2004. (Subseries LNAI).

Vol. 3212: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 862 pages. 2004.

Vol. 3211: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 880 pages. 2004.

Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic. XI, 520 pages. 2004.

Vol. 3209: B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (Eds.), Web Mining: From Web to Semantic Web. IX, 201 pages. 2004. (Subseries LNAI).

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3204: C.A. Peña Reyes, Coevolutionary Fuzzy Modeling. XIII, 129 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Knowledge Discovery in Databases: PKDD 2004. XIX, 560 pages. 2004. (Subseries LNAI).

Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004. XVIII, 580 pages. 2004. (Subseries LNAI).

Vol. 3199: H. Schepers (Ed.), Software and Compilers for Embedded Systems. X, 259 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3196: C. Stary, C. Stephanidis (Eds.), User-Centered Interaction Paradigms for Universal Access in the Information Society. XII, 488 pages. 2004.

Vol. 3195: C.G. Puntonet, A. Prieto (Eds.), Independent Component Analysis and Blind Signal Separation. XXIII, 1266 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), Computer Security – ESORICS 2004. X, 457 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004. (Subseries LNAI).

Vol. 3191: M. Klusch, S. Ossowski, V. Kashyap, R. Unland (Eds.), Cooperative Information Agents VIII. XI, 303 pages. 2004. (Subseries LNAI).

Vol. 3190: Y. Luo (Ed.), Cooperative Design, Visualization, and Engineering. IX, 248 pages. 2004.

Vol. 3189: P.-C. Yew, J. Xue (Eds.), Advances in Computer Systems Architecture. XVII, 598 pages. 2004.

Vol. 3188: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. VIII, 373 pages. 2004.

Vol. 3187: G. Lindemann, J. Denzinger, I.J. Timm, R. Unland (Eds.), Multiagent System Technologies. XIII, 341 pages. 2004. (Subseries LNAI).

Vol. 3186: Z. Bellahsène, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), Database and XML Technologies. X, 235 pages. 2004.

Vol. 3185: M. Bernardo, F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems. VII, 295 pages. 2004.

Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), Trust and Privacy in Digital Business. XI, 299 pages. 2004.

Vol. 3183: R. Traunmüller (Ed.), Electronic Government. XIX, 583 pages. 2004.

Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), E-Commerce and Web Technologies. XI, 370 pages. 2004.

Vol. 3181: Y. Kambayashi, M. Mohania, W. Wöß (Eds.), Data Warehousing and Knowledge Discovery. XIV, 412 pages. 2004.

此为试读,需要完整PDF请访问: www.ertongbook.com

Table of Contents

Part 1. Architectures for Dependability

Part 2. Fault Tolerance in Software Architectures

Part 3. Dependability Analysis in Software Architectures

Part 4. Industrial Experiences

Architectures for Dependability

Enhancing Dependability Through Flexible Adaptation to Changing Requirements*

Michel Wermelinger[1], Georgios Koutsoukos[2], Hugo Lourenço[2], Richard Avillez[2], João Gouveia[2], Luís Andrade[2], and José Luiz Fiadeiro[3]

[1] Dep. de Informática, Univ. Nova de Lisboa, 2829-516 Caparica, Portugal
mw@di.fct.unl.pt
[2] ATX Software SA, Alameda António Sérgio, 7, 1C
2795-023 Linda-a-Velha, Portugal
{firstname.lastname}@atxsoftware.com
[3] Dep. of Computer Science, Univ. of Leicester, Leicester LE1 7RH, UK
jose@fiadeiro.org

Abstract. This paper describes an architectural approach that facilitates the dynamic adaptation of systems to changing domain rules. The approach relies on "coordination contracts", a modelling and implementation primitive we have developed for run-time reconfiguration. Our framework includes an engine that, whenever a service is called, checks the domain rules that are applicable and configures the response of the service before proceeding with the call.

This approach enhances dependability in two essential ways: on the one hand, it guarantees that system execution is always consistent with the domain logic because service response is configured automatically (i.e., without any need for programmer intervention); on the other hand, it makes it possible for changes to be incorporated into existing domain rules, and from new rules to be created, with little effort, because coordination contracts can be superposed dynamically without having to change neither the client nor the service code.

Our approach is illustrated through a case study in financial systems, an area in which dependability arises mainly in the guise of business concerns like adherence to agreed policies and conditions negotiated on a case-by-case basis. We report on an information system that ATX Software developed for a company specialised in recovering bad credit. We show in particular how, by using this framework, we have devised a way of generating rule-dependent SQL code for batch-oriented services.

1 Introduction

This paper describes an architectural approach to system development that facilitates adaptation to change so that organisations can effectively depend on a continued service that satisfies evolving business requirements. This approach has been used in a real project in which ATX Software developed an information

* This paper is a considerably extended version of [1].

R. de Lemos et al. (Eds.): Architecting Dependable Systems II, LNCS 3069, pp. 3–24, 2004.
© Springer-Verlag Berlin Heidelberg 2004

system for a company specialised in recovering bad credit. The approach is based on two key mechanisms:

- the externalisation of the domain rules from the code that implements core system functionalities;
- the encapsulation of the code that enforces those domain rules into so-called coordination contracts that can be created and deleted at run-time, hence adapting computational services to the context in which they are called.

In the concrete case study that we present, the domain rules define the dependency of the recovery process on business concerns of the financial institution and product (e.g., house mortgage) for which the debt is being recovered. At any given time, this business configuration defines the context in which services are called.

These two mechanisms are aimed at two different classes of stakeholders. Domain rules are intended for system users, who have no technical knowledge, so that they can adapt the system in order to cope with requirements of newly or already integrated financial institutions or products. Coordination contracts are intended for system developers to add new behaviour without changing the original service implementation. This is made possible with the ability of coordination contracts to superpose, at run-time, new computations on the services that are being execute locally in system components.

Coordination contracts [2] are a modelling and implementation primitive that allows transparent interception of method calls and as such interfere with the execution of the service in the client. Transparent means that neither the service nor its client are aware of the existence of the coordination contract. Hence, if the system has to be evolved to handle the requirements imposed by new institutions or products, many of the changes can be achieved by parameterising the service (data changes) and by superposing new coordination contracts (behaviour changes), without changing the service's nor the client's code. This was used, for instance, to replace the default calculation of the debt's interest by a different one. The user may then pick one of the available calculation formulae (i.e., coordination contracts) when defining a domain rule.

To be more precise, a coordination contract is applicable to one or more objects (called the contract's participants) and has one or more coordination rules, each one indicating which method of which participant will be intercepted, under which conditions, and what actions to take in that case. In the particular case of the system that we are reporting in this paper, all coordination contracts are unary, the participant being the service affected by the domain rule to which the coordination contract is associated. Moreover, each contract has a single rule. We could have joined all coordination rules that *may be* applicable to the same service into a single contract, but that would be less efficient in run-time and more complex in design time due to more intricate rule definitions. The reason is that once a contract is in place, it will intercept *all* methods given in all the contract's rules, and thus the rule conditions would have to check at run-time if the rule is really applicable, or if the contract was put in place because of another coordination rule.

此为试读,需要完整PDF请访问: www.ertongbook.com