Journal Subline

LNCS 4640

Transactions on Aspect-Oriented Software Development IV

Awais Rashid · Mehmet Aksit
Editors-in-Chief

Springer

Awais Rashid Mehmet Aksit (Eds.)

Transactions on Aspect-Oriented Software Development IV

Springer

Volume Editors

Awais Rashid
Lancaster University
Computing Department
Lancaster LA1 4WA, UK
E-mail: awais@comp.lancs.ac.uk

Mehmet Aksit
University of Twente
Department of Computer Science
Enschede, The Netherlands
E-mail: aksit@ewi.utwente.nl

Library of Congress Control Number: 2007939971

CR Subject Classification (1998): D.2, D.3, I.6, H.4, K.6

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 1861-3027
ISBN-10 3-540-77041-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-77041-1 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

springer.com

© Springer-Verlag Berlin Heidelberg 2007
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12197705 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4640

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
University of California, Los Angeles, CA, 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

Editorial

Volume IV of Transactions on Aspect-Oriented Software Development continues the special issue on Early Aspects from volume III. The special issue was guest edited by João Araújo and Elisa Baniassad and handled by one of the co-editors-in-chief, Mehmet Aksit. The papers in volume III discussed topics pertaining to analysis, visualisation, conflict identification and composition of Early Aspects. The papers in this volume focus on mapping of Early Aspects across the software lifecycle. Complementing this focus on aspect mapping is a special section on Aspects and Software Evolution guest edited by Walter Cazzola, Shigeru Chiba and Gunter Saake—the co-editor-in-chief handling this issue was Awais Rashid.

We wish to thank the guest editors for their commitment and effort in producing such a high quality volume. We also thank the editorial board for their continued guidance, commitment and input on the policies of the journal, the choice of special issues as well as associate-editorship of submitted articles. Thanks are also due to the reviewers who volunteered time amidst their busy schedules to help realize this volume. Most importantly, we wish to thank the authors who have submitted papers to the journal so far, for their contributions maintain the high quality of Transactions on AOSD.

There are two other special issues on the horizon. One focuses on *aspects, dependencies and interactions* (guest editors: Ruzanna Chitchyan, Johan Fabry, Shmuel Katz and Arend Rensink) for which the call for papers has closed and the papers are currently in the review phase. There is an open call for papers for a special issue on *aspects and model-driven engineering* (guest editors: Jean-Marc Jezequel and Robert France). The call will close on 15 November 2007. These special issues coupled with the regular submissions to the journal mean that we have a number of exciting papers to look forward to in future volumes of Transactions on AOSD.

There are also important changes afoot at the journal. At the last editorial board meeting Don Batory and Dave Thomas volunteered to step down from the editorial board. Their input and guidance were invaluable in the start-up period of the journal. Don was also a very active and conscientious associate editor. We thank them both for their contributions. At the same time, we welcome Bill Harrison, Oege de Moor and Shriram Krishnamurthi to the editorial board and look forward to working with them.

Another major change involves the co-editors-in-chief. As per the journal policy, one of the founding co-editors-in-chief, Mehmet Aksit, is stepping down after the first two years of the journal. So this is the last volume Mehmet will be co-editing in this role. Needless to say, Mehmet has been instrumental in the successful launch of the journal and its operations to date and the editorial board is most grateful for his efforts and contributions. We do not lose Mehmet although as he will remain on the editorial board and continue to guide us.

At the same time, it is with great pleasure we welcome Harold Ossher who will be taking over from Mehmet as co-editor-in-chief. Harold's name needs no introduction in the AOSD and software engineering communities. His work on subject-oriented

此为试读,需要完整PDF请访问: www.ertongbook.com

programming laid the early foundations of AOSD and, subsequently, his work on multi-dimensional separation of concerns has been fundamental in influencing how we perceive the notion of aspects. The journal will continue to flourish under his guidance and leadership and we feel that the future for both the journal and the AOSD community at large is very bright.

Awais Rashid and Mehmet Aksit
Co-editors-in-chief

Organization

Editorial Board

Mehmet Aksit, University of Twente
Shigeru Chiba, Tokyo Institute of Technology
Siobhán Clarke, Trinity College Dublin
Theo D'Hondt, Vrije Universtiteit Brussel
Robert Filman, Google
Bill Harrison, Trinity College Dublin
Shmuel Katz, Technion-Israel Institute of Technology
Shriram Krishnamurthi, Brown University
Gregor Kiczales, University of British Columbia
Karl Lieberherr, Northeastern University
Mira Mezini, University of Darmstadt
Oege de Moor, University of Oxford
Ana Moreira, New University of Lisbon
Linda Northrop, Software Engineering Institute
Harold Ossher, IBM Research
Awais Rashid, Lancaster University
Douglas Schmidt, Vanderbilt University

List of Reviewers

Jonathan Aldrich
João Araújo
Don Batory
Klaas van den Berg
Lodewijk Bergmans
Jean Bézivin
Gordon Blair
Johan Brichau
Shigeru Chiba
Ruzanna Chitchyan
Paul Clements
Yvonne Coady
Arie van Deursen
Erik Ernst
Robert Filman
Lidia Fuentes
Alessandro Garcia
Sudipto Ghosh
Jeff Gray

Michael Haupt
Bill Harrison
Robert Hirschfeld
Jean-Marc Jézéquel
Gregor Kiczales
Günter Kniesel
Thomas Ledoux
Cristina Lopes
Roberto Lopez-Herrejon
David Lorenz
Hidehiko Masuhara
Ana Moreira
Klaus Ostermann
Martin Robillard
Americo Sampaio
Christa Schwannninger
Dominik Stein
Mario Südholt
Eric Tanter
Gabriele Tänzer
Peri Tarr
Bedir Tekinerdogan
Emiliano Tramontana
Ian Welch
Jon Whittle

Lecture Notes in Computer Science

Sublibrary 2: Programming and Software Engineering

For information about Vols. 1–4204
please contact your bookseller or Springer

Vol. 4834: R. Cerqueira, R.H. Campbell (Eds.), Middleware 2007. XIII, 451 pages. 2007.

Vol. 4824: A. Paschke, Y. Biletskiy (Eds.), Advances in Rule Interchange and Applications. XIII, 243 pages. 2007.

Vol. 4807: Z. Shao (Ed.), Programming Languages and Systems. XI, 431 pages. 2007.

Vol. 4799: A. Holzinger (Ed.), HCI and Usability for Medicine and Health Care. XVI, 458 pages. 2007.

Vol. 4789: M. Butler, M.G. Hinchey, M.M. Larrondo-Petrie (Eds.), Formal Methods and Software Engineering. VIII, 387 pages. 2007.

Vol. 4767: F. Arbab, M. Sirjani (Eds.), International Symposium on Fundamentals of Software Engineering. XIII, 450 pages. 2007.

Vol. 4764: P. Abrahamsson, N. Baddoo, T. Margaria, R. Messnarz (Eds.), Software Process Improvement. XI, 225 pages. 2007.

Vol. 4762: K.S. Namjoshi, T. Yoneda, T. Higashino, Y. Okamura (Eds.), Automated Technology for Verification and Analysis. XIV, 566 pages. 2007.

Vol. 4758: F. Oquendo (Ed.), Software Architecture. XVI, 340 pages. 2007.

Vol. 4757: F. Cappello, T. Herault, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XVI, 396 pages. 2007.

Vol. 4753: E. Duval, R. Klamma, M. Wolpers (Eds.), Creating New Learning Experiences on a Global Scale. XII, 518 pages. 2007.

Vol. 4749: B.J. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Service-Oriented Computing – ICSOC 2007. XIX, 629 pages. 2007.

Vol. 4748: K. Wolter (Ed.), Formal Methods and Stochastic Models for Performance Evaluation. X, 301 pages. 2007.

Vol. 4741: C. Bessière (Ed.), Principles and Practice of Constraint Programming – CP 2007. XV, 890 pages. 2007.

Vol. 4735: G. Engels, B. Opdyke, D.C. Schmidt, F. Weil (Eds.), Model Driven Engineering Languages and Systems. XV, 698 pages. 2007.

Vol. 4716: B. Meyer, M. Joseph (Eds.), Software Engineering Approaches for Offshore and Outsourced Development. X, 201 pages. 2007.

Vol. 4680: F. Saglietti, N. Oster (Eds.), Computer Safety, Reliability, and Security. XV, 548 pages. 2007.

Vol. 4670: V. Dahl, I. Niemelä (Eds.), Logic Programming. XII, 470 pages. 2007.

Vol. 4652: D. Georgakopoulos, N. Ritter, B. Benatallah, C. Zirpins, G. Feuerlicht, M. Schoenherr, H.R. Motahari-Nezhad (Eds.), Service-Oriented Computing ICSOC 2006. XVI, 201 pages. 2007.

Vol. 4640: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development IV. IX, 191 pages. 2007.

Vol. 4634: H. Riis Nielson, G. Filé (Eds.), Static Analysis. XI, 469 pages. 2007.

Vol. 4620: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development III. IX, 201 pages. 2007.

Vol. 4615: R. de Lemos, C. Gacek, A. Romanovsky (Eds.), Architecting Dependable Systems IV. XIV, 435 pages. 2007.

Vol. 4610: B. Xiao, L.T. Yang, J. Ma, C. Muller-Schloer, Y. Hua (Eds.), Autonomic and Trusted Computing. XVIII, 571 pages. 2007.

Vol. 4609: E. Ernst (Ed.), ECOOP 2007 – Object-Oriented Programming. XIII, 625 pages. 2007.

Vol. 4608: H.W. Schmidt, I. Crnković, G.T. Heineman, J.A. Stafford (Eds.), Component-Based Software Engineering. XII, 283 pages. 2007.

Vol. 4591: J. Davies, J. Gibbons (Eds.), Integrated Formal Methods. IX, 660 pages. 2007.

Vol. 4589: J. Münch, P. Abrahamsson (Eds.), Product-Focused Software Process Improvement. XII, 414 pages. 2007.

Vol. 4574: J. Derrick, J. Vain (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2007. XI, 375 pages. 2007.

Vol. 4556: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part III. XXII, 1020 pages. 2007.

Vol. 4555: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part II. XXII, 1066 pages. 2007.

Vol. 4554: C. Stephanidis (Ed.), Universal Acess in Human Computer Interaction, Part I. XXII, 1054 pages. 2007.

Vol. 4553: J.A. Jacko (Ed.), Human-Computer Interaction, Part IV. XXIV, 1225 pages. 2007.

Vol. 4552: J.A. Jacko (Ed.), Human-Computer Interaction, Part III. XXI, 1038 pages. 2007.

Vol. 4551: J.A. Jacko (Ed.), Human-Computer Interaction, Part II. XXIII, 1253 pages. 2007.

Vol. 4550: J.A. Jacko (Ed.), Human-Computer Interaction, Part I. XXIII, 1240 pages. 2007.

Vol. 4542: P. Sawyer, B. Paech, P. Heymans (Eds.), Requirements Engineering: Foundation for Software Quality. IX, 384 pages. 2007.

Vol. 4536: G. Concas, E. Damiani, M. Scotto, G. Succi (Eds.), Agile Processes in Software Engineering and Extreme Programming. XV, 276 pages. 2007.

Vol. 4530: D.H. Akehurst, R. Vogel, R.F. Paige (Eds.), Model Driven Architecture - Foundations and Applications. X, 219 pages. 2007.

Vol. 4523: Y.-H. Lee, H.-N. Kim, J. Kim, Y.W. Park, L.T. Yang, S.W. Kim (Eds.), Embedded Software and Systems. XIX, 829 pages. 2007.

Vol. 4498: N. Abdennahder, F. Kordon (Eds.), Reliable Software Technologies - Ada-Europe 2007. XII, 247 pages. 2007.

Vol. 4486: M. Bernardo, J. Hillston (Eds.), Formal Methods for Performance Evaluation. VII, 469 pages. 2007.

Vol. 4470: Q. Wang, D. Pfahl, D.M. Raffo (Eds.), Software Process Dynamics and Agility. XI, 346 pages. 2007.

Vol. 4468: M.M. Bonsangue, E.B. Johnsen (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 317 pages. 2007.

Vol. 4467: A.L. Murphy, J. Vitek (Eds.), Coordination Models and Languages. X, 325 pages. 2007.

Vol. 4454: Y. Gurevich, B. Meyer (Eds.), Tests and Proofs. IX, 217 pages. 2007.

Vol. 4444: T. Reps, M. Sagiv, J. Bauer (Eds.), Program Analysis and Compilation, Theory and Practice. X, 361 pages. 2007.

Vol. 4440: B. Liblit, Cooperative Bug Isolation. XV, 101 pages. 2007.

Vol. 4408: R. Choren, A. Garcia, H. Giese, H.-f. Leung, C. Lucena, A. Romanovsky (Eds.), Software Engineering for Multi-Agent Systems V. XII, 233 pages. 2007.

Vol. 4406: W. De Meuter (Ed.), Advances in Smalltalk. VII, 157 pages. 2007.

Vol. 4405: L. Padgham, F. Zambonelli (Eds.), Agent-Oriented Software Engineering VII. XII, 225 pages. 2007.

Vol. 4401: N. Guelfi, D. Buchs (Eds.), Rapid Integration of Software Engineering Techniques. IX, 177 pages. 2007.

Vol. 4385: K. Coninx, K. Luyten, K.A. Schneider (Eds.), Task Models and Diagrams for Users Interface Design. XI, 355 pages. 2007.

Vol. 4383: E. Bin, A. Ziv, S. Ur (Eds.), Hardware and Software, Verification and Testing. XII, 235 pages. 2007.

Vol. 4379: M. Südholt, C. Consel (Eds.), Object-Oriented Technology. VIII, 157 pages. 2007.

Vol. 4364: T. Kühne (Ed.), Models in Software Engineering. XI, 332 pages. 2007.

Vol. 4355: J. Julliand, O. Kouchnarenko (Eds.), B 2007: Formal Specification and Development in B. XIII, 293 pages. 2006.

Vol. 4354: M. Hanus (Ed.), Practical Aspects of Declarative Languages. X, 335 pages. 2006.

Vol. 4350: M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, All About Maude - A High-Performance Logical Framework. XXII, 797 pages. 2007.

Vol. 4348: S. Tucker Taft, R.A. Duff, R.L. Brukardt, E. Plödereder, P. Leroy, Ada 2005 Reference Manual. XXII, 765 pages. 2006.

Vol. 4346: L. Brim, B.R. Haverkort, M. Leucker, J. van de Pol (Eds.), Formal Methods: Applications and Technology. X, 363 pages. 2007.

Vol. 4344: V. Gruhn, F. Oquendo (Eds.), Software Architecture. X, 245 pages. 2006.

Vol. 4340: R. Prodan, T. Fahringer, Grid Computing. XXIII, 317 pages. 2007.

Vol. 4336: V.R. Basili, H.D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, R.W. Selby (Eds.), Empirical Software Engineering Issues. XVII, 193 pages. 2007.

Vol. 4326: S. Göbel, R. Malkewitz, I. Iurgel (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. X, 384 pages. 2006.

Vol. 4323: G. Doherty, A. Blandford (Eds.), Interactive Systems. XI, 269 pages. 2007.

Vol. 4322: F. Kordon, J. Sztipanovits (Eds.), Reliable Systems on Unreliable Networked Platforms. XIV, 317 pages. 2007.

Vol. 4309: P. Inverardi, M. Jazayeri (Eds.), Software Engineering Education in the Modern Age. VIII, 207 pages. 2006.

Vol. 4294: A. Dan, W. Lamersdorf (Eds.), Service-Oriented Computing – ICSOC 2006. XIX, 653 pages. 2006.

Vol. 4290: M. van Steen, M. Henning (Eds.), Middleware 2006. XIII, 425 pages. 2006.

Vol. 4279: N. Kobayashi (Ed.), Programming Languages and Systems. XI, 423 pages. 2006.

Vol. 4262: K. Havelund, M. Núñez, G. Roşu, B. Wolff (Eds.), Formal Approaches to Software Testing and Runtime Verification. VIII, 255 pages. 2006.

Vol. 4260: Z. Liu, J. He (Eds.), Formal Methods and Software Engineering. XII, 778 pages. 2006.

Vol. 4257: I. Richardson, P. Runeson, R. Messnarz (Eds.), Software Process Improvement. XI, 219 pages. 2006.

Vol. 4242: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development II. IX, 289 pages. 2006.

Vol. 4229: E. Najm, J.-F. Pradat-Peyre, V.V. Donzeau-Gouge (Eds.), Formal Techniques for Networked and Distributed Systems - FORTE 2006. X, 486 pages. 2006.

Vol. 4227: W. Nejdl, K. Tochtermann (Eds.), Innovative Approaches for Learning and Knowledge Sharing. XVII, 721 pages. 2006.

Vol. 4218: S. Graf, W. Zhang (Eds.), Automated Technology for Verification and Analysis. XIV, 540 pages. 2006.

Vol. 4214: C. Hofmeister, I. Crnković, R. Reussner (Eds.), Quality of Software Architectures. X, 215 pages. 2006.

Table of Contents

Focus: Early Aspects – Mapping Across the Lifecycle

Focus: Aspects and Software Evolution

此为试读，需要完整PDF请访问：www.ertongbook.com

Guest Editors' Introduction: Early Aspects — Mapping Across the Lifecycle

João Araújo[1] and Elisa Baniassad[2]

[1] Universidade Nova de Lisboa, Portugal
ja@di.fct.unl.pt
[2] Chinese University of Hong Kong, China
elisa@cse.cuhk.edu.hk

Early Aspects are aspects found in the early life-cycle phases of software development, including requirements elicitation and analysis, domain analysis and architecture design activities. Aspects at these stages crosscut the modular units appropriate for their lifecycle activity; traditional requirements documentation, domain knowledge capture and architectural artifacts do not afford separate description of early aspects. As such, early aspects necessitate new modularizations to be effectively captured and maintained. Without new tools and techniques, early aspects remain tangled and scattered in life-cycle artifacts, and may lead to development, maintenance and evolution difficulties.

Overview of the Articles and the Evaluation Process: This special issue consists of eight articles, selected out of ten submissions. Each were evaluated by three reviewers and revised at least twice over a period of 7 months.

The Early Aspects special issue covers three main areas of research, and is split over two volumes of the journal. The papers in vol. III focused on Analysis and Visualization, and Conflicts and Composition. This volume contains papers on mapping early aspects throughout the life-cycle.

Mapping

The relationship between aspects between life-cycle phases is of primary interest to the Early Aspects community. In this work, researchers attempt to draw a correspondence between concerns in one lifecycle phase, to those found in another. These approaches may involve link recovery, in which existing artifacts are examined and the links between them derived, link formation, in which aspects in each phase are captured in such a way that promotes traceability between them, or link exploitation, in which traceability links are made explicit, and then exploited for other purposes. Here we present two papers related to mapping between aspects at life-cycle phases.

COMPASS: Composition-Centric Mapping of Aspectual Requirements to Architecture by *Ruzanna Chitchyan, Mónica Pinto, Awais Rashid and Lidia Fuentes*

This paper presents COMPASS, an approach that offers a systematic means to derive an aspect-oriented architecture from a given aspect-oriented requirements specification. COMPASS provides an aspect-oriented requirements description

A. Rashid and M. Aksit (Eds.): Transactions on AOSD IV, LNCS 4640, pp. 1–2, 2007.
© Springer-Verlag Berlin Heidelberg 2007

language (RDL) that enriches the informal natural language requirements with additional compositional information. COMPASS also offers an aspect-oriented architecture description language (AO-ADL) that uses components and connectors as the basic structural elements with aspects treated as specific types of components.

Aspects at the Right Time by *Pablo Sánchez, Lidia Fuentes, Andrew Jackson and Siobhán Clarke*

This paper describes an aspect mapping from requirements (specified in Theme/Doc) to architecture (specified in CAM) to design (specified in Theme/UML). The mapping includes heuristics to guide to the right time to specify the right aspect properties. Moreover, it allows aspect decisions captured at each stage to be refined at later stages as appropriate. Also, they provide a means to record decisions that capture the alternatives considered and the decision justification, crucial for managing aspect evolution at the right time.

COMPASS: Composition-Centric Mapping of Aspectual Requirements to Architecture

Ruzanna Chitchyan[1], Mónica Pinto[2], Awais Rashid[1], and Lidia Fuentes[2]

[1] Computing Department, Lancaster University, Lancaster LA1 4WA, UK
{rouza, marash}@comp.lancs.ac.uk
[2] Dept. Lenguajes y Ciencias de la Computación, University of Málaga, Málaga, Spain
{pinto, lff}@lcc.uma.es

Abstract. Currently there are several approaches available for aspect-oriented requirements engineering and architecture design. However, the relationship between aspectual requirements and architectural aspects is poorly understood. This is because aspect-oriented requirements engineering approaches normally extend existing requirements engineering techniques. Although this provides backward compatibility, the composition semantics of the aspect-oriented extension are limited by those of the approaches being extended. Consequently, there is limited or no knowledge about how requirements-level aspects and their compositions map on to architecture-level aspects and architectural composition. In this paper, we present COMPASS, an approach that offers a systematic means to derive an aspect-oriented architecture from a given aspect-oriented requirements specification. COMPASS is centred on an aspect-oriented requirements description language (RDL) that enriches the usual informal natural language requirements with additional compositional information derived from the semantics of the natural language descriptions themselves. COMPASS also offers an aspect-oriented architecture description language (AO-ADL) that uses components and connectors as the basic structural elements (similar to traditional ADLs) with aspects treated as specific types of components. Lastly, COMPASS provides a set of concrete mapping guidelines, derived from a detailed case study, based on mapping patterns of compositions and dependencies in the RDL to patterns of compositions and dependencies in the AO-ADL. The mapping patterns are supported via a structural mapping of the RDL and AO-ADL meta-models.

Keywords: aspect–oriented software development, early aspects, requirements engineering, architecture design, requirements to architecture mapping, requirements composition, architecture composition.

1 Introduction

As aspect-oriented software development (AOSD) grows in popularity, more and more requirements [9, 45, 54, 69] and architecture [9, 67] level approaches emerge. They all aim to improve modular representation and analysis of crosscutting concerns at the requirements- or architecture-level, but no single approach covers both activities: starting from requirements and resulting in an architecture specification for

A. Rashid and M. Aksit (Eds.): Transactions on AOSD IV, LNCS 4640, pp. 3 – 53, 2007.
© Springer-Verlag Berlin Heidelberg 2007

the given requirements. Though some approaches, e.g., [45, 54], provide initial insights into architectural choices, no concrete mapping guidelines for deriving the architecture are provided. Our approach, COMPASS, is based on a composition-centric perspective for such requirements-to-architecture mapping. That is, it focuses on the compositional information and dependencies of concerns at the requirements-level and utilises these as a basis for a systematic transition from an aspect-oriented requirements specification to an aspect-oriented architecture. Compositions are the embodiments of aspectual interactions in requirements. The mapping facilitated by COMPASS allows a developer to utilise requirement compositions to pinpoint the likely aspectual relationships in architecture that originate from the requirements.[1]

Such a composition-centric approach requires rich composition semantics at the requirements-level. However, the majority of current aspect-oriented requirements engineering (AORE) techniques have been developed as extensions to other contemporary requirements engineering (RE) approaches. For instance, the AORE with ARCADE approach [54] extends a viewpoint-based requirements engineering model called PREView [64] with the notion of aspects. Similarly, the aspect-oriented use case approach [31] extends the traditional use case model with aspectual use cases. Although this provides backward compatibility in terms of software processes and development practices, it also restricts these AO approaches to the same dominant decomposition as the extended RE approach, turning everything that does not fit quite well with the base[2] approach into aspects. The semantics of such concerns put into this "aspect-bin" are often under-investigated; they frequently do not receive adequate representation and reasoning support either. Though some of these concerns may very well align with the given notations (often adopted from the base approach, or new dedicated "add-ons") and classification, others may be forced into such adapted frameworks. For instance, in case of aspectual use cases [31] the *extend* and *insert* use cases are re-classified as "aspectual" and an additional set of *infrastructure* use cases is introduced for the representation of non-functional concerns. Although the *extend* and *insert* use cases fit very well into the traditional use case (i.e., functionality-related) semantics and representation, the *infrastructure* use cases are forced to "functionalise" the qualitative semantics of non-functional concerns. As such the expressive and compositional power of the aspect-oriented approach is limited by that of the base approach.

The provision of richer composition semantics at the requirements-level is the first aim of COMPASS. The COMPASS Requirements Description Language (RDL) partitions requirements into concerns like most RE techniques but with two main differences. First, it takes a symmetric approach to such partitioning, i.e., aspects and base concerns are treated uniformly using the same abstraction, a *concern*. [46, 66]. Second, it enriches the usual informal natural language requirements with additional compositional information derived from the semantics of the natural language descriptions themselves. This compositional information is utilised for semantics-based composition of requirements-level concerns. It also provides core insights into

[1] It must be noted that other aspects, motivated by the solution domain, may also arise in architecture. Such solution domain aspects are not targeted by this approach. Our compositions pinpoint the aspects arising from the problem domain, i.e., the requirements.

[2] "Base approach" here is the approach being extended with Aspects.

the intentionality of a requirement hence facilitating a clearer mapping to relevant architectural elements. The natural language requirements' annotation with the RDL is fully automated via our Wmatrix [58] natural language processor. Tool support is also available for crosscutting concern identification [56, 57].

A composition-centric approach also requires clearer architectural composition semantics for aspects. Presently in many cases, aspect-oriented architecture design approaches adopt concepts introduced by aspect-oriented programming (AOP) languages, without questioning how appropriate these may be at the architecture level. Some of the examples of such programming language driven features are: introductions; asymmetric representation — i.e., use of different artefacts for base functionality and aspectual behaviour; and the lack of separation of compositional information (i.e., the pointcuts) from the aspect behaviour (i.e., the advice). Although such features provide a closer alignment between architecture and a given AOP language, they do not always help to capture the fundamental nature of software architecture descriptions, unnecessarily complicating architecture comprehensibility and evolution. For instance, AOP `introductions` are implementation-specific mechanisms thought to extend the interface and behaviour of a class when only the binary code is available. This is not appropriate at the architecture level, where instead the interface of a component should be extended by transforming the component into a composite component with multiple interfaces. Also, pointcuts specify composition of architectural components, be it aspectual ones, and, therefore, ought to be part of the connector semantics rather than be included within the aspect specification.

The provision of suitable abstraction and composition mechanisms at the architecture-level is the second aim of COMPASS. We propose an aspect-oriented ADL (AO-ADL) based on a symmetric decomposition model — it uses components and connectors as the basic structural elements (similar to traditional ADLs) with aspects treated as specific types of components. Connectors are enriched with additional composition semantics to cope with the crosscutting effect of aspectual components.

Having enriched requirements and architecture models with suitable aspect composition semantics, COMPASS provides a set of concrete mapping guidelines, derived from a detailed case study, based on mapping patterns of compositions and dependencies in the RDL to patterns of compositions and dependencies in the AO-ADL. The mapping patterns are supported via a structural mapping of the RDL and AO-ADL meta-models.

The mapping guidelines in COMPASS are a significant contribution not only to improving transparency of transition from aspectual requirements to architecture but also to the general issue of relating requirements to architecture in a systematic manner. The third goal of COMPASS is to establish clear links between the requirements-level aspects and their compositions with architecture elements and transition from the requirements-level to architecture. The architecture derived from COMPASS mappings acts as a starting point for refinement and elaboration into an architectural solution. We leave the topics of architecture refinement and elaboration out of this paper for a separate discussion, and focus on the actual mappings themselves.

The rest of the paper is structured as follows. Section 2 presents how COMPASS fits within the software development activities. Section 3 discusses our RDL and its composition semantics. Section 4 discusses the abstraction and composition mechanisms in our AO-ADL. Section 5 presents the mapping patterns based on the compositional information and dependencies as well as the structural mapping between the RDL and AO-ADL meta-models. The discussion in Sects. 3,4 and 5 is based on a concrete case study of an online auction system that has also been used as a basis for eliciting the mapping guidelines. Section 6 discusses and demonstrates application of guidelines, as well as presents some difficult issues that the COMPASS approach faces. Section 7 discusses related work and Sect. 8 concludes the paper. The summary of the mapping guidelines is presented in Appendix A.

2 COMPASS Within the Software Development Process

In order to explain the relation of COMPASS to the general software development activities, we have highlighted the COMPASS activities in Fig. 1. As shown, COMPASS is concerned with the link between the RE and Architectural activities, which is represented as the oval containing "Requirements to Architecture Mapping" activity along with its adjacent arrows.

Figure 1 explicitly mentions a number of RE activities that come before the COMPASS activities, as well as a number of Architecture Design activities that come after. Though none of these pre and post COMPASS activities are the focus of this paper, for the sake of completeness and clarity, we provide a brief overview of some related issues in this section.

2.1 Requirements Engineering

For use of COMPASS we do not prescribe any particular RE technique, the only specified condition being that the used technique will contain natural language requirements specification. The decision to use natural language specification was motivated mainly by the fact that the majority of the requirements are still specified in natural language text [12]. Clearly, it is hardly possible to establish a direct mapping from raw natural language text to architecture design. Thus, a number of tasks dedicated to concern identification and structuring requirements into a specification document need to be undertaken. The specification document will then form input for the COMPASS mapping.

Thus, any kind of structured text-based specification (such as viewpoints or use cases) can form COMPASS input.[3] Our own work on producing such structured natural language requirements for AORE from initial natural language (NL) text is presented in [56, 57]. In brief, our approach for structuring is based on use of tool-supported corpus-based natural language processing methodology. We initially apply an NL processor (called WMATRIX [6, 58]), which helps to identify the main topics of interest in the given natural language document by comparing the given document

[3] Input for COMPASS can be produced by any other RE approach that uses natural language. It is not necessary to use specifically our approach that is discussed further in this section.

此为试读，需要完整PDF请访问：www.ertongbook.com