LNCS 4620 Journal Subline

Transactions on Aspect-Oriented Software Development III

Awais Rashid · Mehmet Aksit
Editors-in-Chief

Springer

Awais Rashid Mehmet Aksit (Eds.)

Transactions on Aspect-Oriented Software Development III

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: 2007939176

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 3-540-75161-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-75161-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12162321 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4620

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

Editorial Board

Editorial

Welcome to Volume III of Transactions on Aspect-Oriented Software Development. Since its launch in 2006 the journal has attracted a steady stream of submissions, both to its indefinitely open call for papers and to calls pertaining to special issues focusing on key topics in AOSD. At the time of writing this editorial, the total number of submissions to the journal stands at 78. This is very healthy given that it is only the second year of the journal.

The journal aims to maintain the highest standards expected of an archival work in AOSD while ensuring timely feedback and notification to the authors. Each paper is handled by an associate editor, appointed by the co-editors-in-chief, who is a specialist on the specific topic. We ensure that the associate editor does not have a conflict of interest relating to the paper assigned to him/her. If the associate editor deems the paper to be worthy of a review, s/he solicits reviews from at least three reviewers on the quality of the work. On the basis of these reviews, s/he then makes a recommendation to reject the paper; ask the authors to resubmit the paper with major revisions; accept the paper with minor revisions or accept it without any further changes. We aim to notify the authors about the outcome of the reviews within 12 weeks. In a small number of cases, either unforeseen circumstances or other commitments of reviewers may lead to some further delay but we are pleased to say that such cases remain a minority.

In cases where major or minor revisions are recommended, the authors are expected to address the reviewers' comments. Papers with major changes are passed on to the reviewers again for a second review. If, even after the second review, a paper cannot be accepted as it is or subject to minor changes, then the paper is rejected. This is to avoid an endless review cycle. The procedure is applied pragmatically in that where there is a significant difference of opinion amongst the reviewers and, provided the associate editor recommends so, the authors are given a third and final opportunity to address the reviewers' comments.

Each special issue is handled by one of the co-editors-in-chief who works closely with the guest editors to ensure that the journal's review process is followed and quality standards are maintained. Given that aspect-oriented software development is a young discipline it was decided at the editorial board meeting in March 2006 that the co-editor-in-chief not handling a special issue should be locked out of the review process completely. This allows such a co-editor-in-chief to author a paper for the special issue as normally co-editors-in-chief cannot submit a paper to the journal.

This volume constitutes the first part of the special issue on Early Aspects guest edited by João Araújo and Elisa Baniassad. The handling co-editor-in-chief was Mehmet Aksit. The special issue was very successful in attracting high quality submissions and, as a result, had to be split over two volumes of the journal. The papers in this volume focus on analysis, visualisation, conflict identification and composition of Early Aspects. The papers in volume IV focus on mapping of Early Aspects across the software lifecycle.

We wish to 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. We also thank the guest editors, João Araújo and Elisa Baniassad, for the excellent job they did with the special issue—the proposal they prepared is now used as a model for all special issue proposals to Transactions on AOSD. 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.

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

Organization

Editorial Board

List of Reviewers

Lecture Notes in Computer Science

Sublibrary 2: Programming and Software Engineering

For information about Vols. 1– 4199
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. 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.

Vol. 4204: F. Benhamou (Ed.), Principles and Practice of Constraint Programming - CP 2006. XVIII, 774 pages. 2006.

Table of Contents

Guest Editors' Introduction: Early Aspects—Analysis, Visualization, Conflicts and Composition

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 lifecycle artifacts, and may lead to development, maintenance and evolution difficulties.

The Early Aspects community has grown significantly since its inception as a workshop at the first conference on Aspect Oriented Software Development in 2001. Since then, the workshop series has flourished, becoming a regular feature of several conferences, and papers presenting and studying new Early Aspects techniques have been published in many major venues. Early aspects research groups now span the globe, and bridge industry and academia.

The level of maturity reached by the Early Aspects work prompted us to edit this special issue on Early Aspects. We believe that this issue will support the cross-fertilization of ideas between those focused on research throughout all phases of the software lifecycle, and will help researchers identify new questions in the world of Early Aspects.

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 seven months.

The Early Aspects special issue covers three main areas of research, and is split over two volumes of the journal. This volume presents papers in the areas of Analysis and Visualization, and Conflicts and Composition. Volume. IV contains papers on mapping early aspects throughout the lifecycle.

1 Analysis and Visualization

Early aspects research often involves examination of existing, traditionally organized, artifacts, and refactoring them into an aspect-oriented organization. This process might encompass identification of aspects in early requirements documents, examination of the relevance of early aspects in a certain type of artifact, how early

A. Rashid and M. Aksit (Eds.): Transactions on AOSD III, LNCS 4620, pp. 1–3, 2007.

aspects can be used to better represent certain early lifecycle documents, or how to more effectively capture aspects for the sake of development activities in the early lifecycle. Early aspects also require new approaches for formation and visualizing lifecycle artifacts. For example, new techniques are required for elicitation and capture of requirements, the maintenance of domain information, and the formation and presentation of architectural information. In this issue, we present four papers which span this area, and touch upon its salient research questions.

EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering *by Américo Sampaio, Awais Rashid, Ruzanna Chitchyan, and Paul Rayson*
This paper describes the EA-Miner tool-based approach, which provides automated support for mining various types of concerns from a variety of early stage requirements documents and how these concepts can be structured into specific aspect-oriented requirements models. The automation consists of natural language processing, to reason about properties of the requirements as well as the utilization of semantics revealed by the natural language analysis in building the models.

Analysis of Early Aspects in Requirements Goal Models: A Concept-Driven Approach *by Nan Niu and Steve Easterbrook*
This paper presents a rigorous approach to conceptual analysis of stakeholder concerns. The authors use the repertory grid technique to identify terminological interference between stakeholders' descriptions of their goals, and formal concept analysis to uncover conflicts and trade-offs between these goals. The approach is applied to the goal models, commonly used in requirements analysis.

Analysis of Crosscutting in Early Software Development Phases based on Traceability *by Klaas van den Berg, José María Conejero, and Juan Hernández*
This paper proposes a conceptual framework for crosscutting where crosscutting is defined in terms of trace relations. The definition of crosscutting is formalized using linear algebra, and represented with matrices and matrix operations. Thus, crosscutting can be clearly distinguished from scattering and tangling. With this definition and transitivity of trace relations, crosscutting can be identified and traced through software development, also in early phases.

Visualizing Early Aspects with Use Case Maps *by Gunter Mussbacher, Daniel Amyot and Michael Weiss*
This paper describes how scenario-based aspects can be modelled at the requirements level unobtrusively and with the same techniques as for non-aspectual systems, with the help of Use Case Maps. These are a visual scenario notation under standardization by the International Telecommunication Union. With Use Case Maps, aspects as well as pointcut expressions are modelled in a visual way which is generally considered the preferred choice for models of a high level of abstraction.

2 Conflicts and Composition

With Early Aspect separation of concerns comes the need for composition of concerns. Early aspects composition is present in all early lifecycle phases.

Mechanisms for weaving early aspects into traditional artifacts are needed. In the requirements phase, composition means both the recombination of separately described requirements, and also the consideration of clashes between the semantics of those requirements. Here we present two papers in this area: one dealing with weaving aspects in design, and the other presenting an approach for handling conflicts in aspectual requirements.

Handling Conflicts in Aspectual Requirements Compositions *by Isabel Sofia Brito, Filipe Vieira, Ana Moreira, and Rita A. Ribeiro*
This paper discusses the use of Multiple Criteria Decision Making methods to support aspectual conflict management in the context of Aspect-Oriented Requirements Engineering. A conflict is detected whenever two or more concerns that contribute negatively to each other and have the same importance need to be composed together. The presented solution relies on the use of the obtained concern rankings to handle unresolved conflicts.

Weaving Multiple Aspects in Models *by Jacques Klein, Franck Fleurey, and Jean-Marc Jézéquel*
This paper presents an approach to statically weave behavioral aspects into sequence diagrams. The weaving process is automated, and takes into account the semantics of the model used, i.e., the partial order that a SD induces. To enable the weaving of multiple aspects, a new interpretation for pointcuts to allow join points to match them more flexibly is proposed.

EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering

Américo Sampaio, Awais Rashid, Ruzanna Chitchyan, and Paul Rayson

Computing Department, InfoLab 21, Lancaster University, Lancaster LA1 4WA, UK
{a.sampaio, awais, rouza, paul}@comp.lancs.ac.uk

Abstract. Aspect-oriented requirements engineering (AORE) provides separation of concerns at the requirements level. In order to cope with concern identification and structuring into different requirements models, tool support is vital to effectively reduce the burden of performing various AORE tasks. This paper describes how the EA-Miner tool-based approach provides automated support for mining various types of concerns from a variety of early stage requirements documents and how these concepts can be structured into specific aspect-oriented requirements models (e.g., viewpoints-based, use-case-based). The key insight for early-stage requirements automation is the use of natural language processing to reason about properties of the requirements as well as the utilization of semantics revealed by the natural language analysis in building the models. Evaluation of EA-Miner shows promising results concerning time-effectiveness and accuracy of undertaking AORE activities and building requirements models. Moreover, an industrial case study conducted at Siemens AG investigated how the tool performs in a real-world setting by analysing what benefits it brings and challenges it faces during AORE analysis. The EA-Miner analysis enabled to find concerns that were considered relevant by a research team at Siemens that is re-implementing the investigated system with aspect-oriented languages. Moreover, the exposure of the tool to industrial requirements written by different developers also revealed some challenges imposed by the structure of the documentation and the different use of vocabulary terms hence providing new paths to explore and improve the tool in the future such as better pre-processing support, "domain synonym" identification and detection of poorly written requirements.

1 Introduction

Requirements engineering (RE) is considered to be a fundamental part of the software engineering lifecycle [1–3] as poor requirements can have a significant impact in later stages of the life cycle and can often be a critical factor in the failure of a project.

One of the initial tasks in RE is gathering the requirements from the end users, managers, and other stakeholders. This is a challenging task since requirements engineers and stakeholders normally have different backgrounds and knowledge about the system under investigation that complicates their communication and understanding of the system goals and uses. Generally, during this process, the requirements engineer somehow records these requirements (e.g., creating a report, generating interview transcripts) to use them later for creating a more detailed specification of the system.

A. Rashid and M. Aksit (Eds.): Transactions on AOSD III, LNCS 4620, pp. 4–39, 2007.

However, in some real scenarios the requirements engineers do not have the opportunity to have much contact with the stakeholders, for example, in mass market application development (e.g., web and off-the-shelf software) [4] where the number and diversity of users can be extremely high. In these cases the requirements engineers have to elicit the requirements based on previous knowledge about the domain or based on available documentation such as marketing studies, legacy specifications and user manuals.

In both cases the goal of the requirements engineers is to make a transition from understanding the "problem world" and creating a requirements specification that represents the system under investigation and that will help developers to build the system in the "solution world" [2, 5]. During this transition process several documents with various structures (e.g., interview transcripts, user manuals, marketing analysis and legacy specifications) can be used to inform the requirements engineer.

Generally this transition process is done manually which can be very time-consuming depending on the size and complexity of the system. For example, consider that the input to the requirements specification and analysis task is a 60 page contract document and a 50 page legacy user manual. Given that the standard average rate of reading is between 250 and 350 words per minute, it is not difficult to realize that reading all the information and transforming it into a more structured RE model demands a huge effort. Therefore, in order to reduce this effort, it is vital to provide automated support.

As most documents used in RE are written in natural language, some researchers [6–14] have found promising results in automating RE tasks by building tools that apply natural language processing (NLP) techniques with requirements-related documents as input in order to automatically identify concepts and build RE models. Applying NLP in requirements interpretation is also challenging as natural language is not as precise as design and implementation languages containing lots of ambiguities and complex semantics.

Regarding structuring of requirements, recently, some researchers [15–19] have proposed the adoption of aspect-oriented requirements engineering (AORE) as an effective approach to achieve separation of concerns. AORE is based on the observation that, similar to what was evidenced by the AOP community, requirements artifacts can contain tangling and scattering that need special treatment. This treatment is provided by adapting current RE approaches with new abstractions (called early aspects) that modularize crosscutting concerns at RE level, thus bringing the following benefits [15–19]:

- Facilitate detection of conflicts between broadly-scoped requirements;
- Simplify analysis of interactions between early aspects (e.g., the impact that security can have on response time);
- Facilitate mapping to later stages (e.g., architecture, design and code) thus providing homogeneity in an aspect-oriented development process.

Even though separation of crosscutting concerns provides benefits, building an AORE specification with existing AORE approaches suffers from the same problems as for classical RE techniques mentioned above. In the case of AORE it is even more challenging since the identification and structuring of requirements level aspects, base abstractions and crosscutting relationships is harder due to these concepts being

scattered and tangled across the documents. Moreover, the fact that AORE is a novel approach complicates the analysis since many system analysts do not have good understanding of early aspects.

This is where the EA-Miner tool-based approach comes into play by offering automated support for identifying the abstractions of different AORE techniques (e.g., viewpoints [20] based, use case [21] based) and helping to build the models. The tool's automated support helps to reduce the time spent to:

- **Identify model abstractions:** For example concepts such as use cases, viewpoints, and early aspects that belong to a specific requirements technique (e.g., Use Case based AORE [22], Viewpoints based AORE [16, 17]) can be automatically mined from different elicitation documents;
- **Structure abstractions into various models:** The tool offers features to edit the identified abstractions (add, remove, filter) as well as to map them into a chosen model (e.g., a structured AORE specification based on viewpoints or use cases).

It is important to mention that EA-Miner's automated support does not replace the work of the requirements engineer but only helps him/her to save time by focusing on key information. The key insight for early-stage requirements automation is the use of natural language processing (NLP) to reason about properties of the requirements as well as the utilization of semantics revealed by the natural language analysis in building the models. The use of NLP techniques to help with AORE automation was initially investigated by our previous work [23, 24] and provided some insights (e.g., which NLP techniques could be used for identifying model concepts) that helped us to reach the current state of the tool's implementation. After this, we have added several features on the tool such as synonym and stemming filtering, frequency analysis, and support for functional crosscutting as well as made several improvements on the identification mechanisms. Moreover, we have conducted several case studies including an industrial case study to evaluate the tool.

Most AORE approaches [16–19, 25] do not provide tool support for the identification of early aspects from requirements documents with the exception of Theme/Doc [15]. Therefore, EA-Miner offers a key contribution to complement these approaches by automating the identification task for them. Moreover, our NLP-based mining analysis and techniques used (Sects. 3,4) offer a higher degree of automation when compared to other mining approaches (Sect. 7) as the input requested from the user is minimal.

The remainder of this paper is structured as follows. Section 2 explains how EA-Miner can be utilized in an AORE process. Section 3 gives an overview of the utilization of natural language techniques for requirements model automation. Section 4 shows how EA-Miner uses these NLP techniques to automate the identification of concepts and mapping of models. Section 5 evaluates the tool showing its time-effectiveness and also presents data regarding the quality of the produced models. Moreover, an industrial case study shows how the tool can perform in a real-world setting and what benefits it can bring for the development process such as identifying relevant concerns that were missed by domain experts. Section 6 provides further discussion of EA-Miner and its features. Section 7 presents an overview of existing related work while Sect. 8 concludes the paper.

2 EA-Miner and the AORE Process

Recently, several researchers have worked on creating new approaches [15, 25, 26] or adapting contemporary requirements engineering approaches (e.g., viewpoints [3], scenarios [21], goals [27]) to what have been called AORE approaches such as Viewpoint-based AORE [16, 17], Scenario-based AORE [18] and goal-based AORE [19].

The common goal of all these AORE approaches is to provide an appropriate separation of concerns at the requirements level modularizing crosscutting properties in early aspects. While some approaches, often categorized as asymmetric, provide a clear separation of what are the base and crosscutting abstractions (e.g., in [16, 17] viewpoints are base abstractions while early aspects[1] are broadly scoped properties, such as security, availability, response time, that crosscut several viewpoints) other approaches, categorized as symmetric, give a uniform treatment to the decomposition units considering everything to be a concern [25].

It is not our intention to get into details of the advantages and disadvantages of each of the AORE approaches as our goal for EA-Miner is to offer a framework (tool + guidelines) that can be used with any AORE approach. In Fig. 1 we show a "general AORE process" which contains a set of activities that are common to most AORE approaches as described in [16–18].

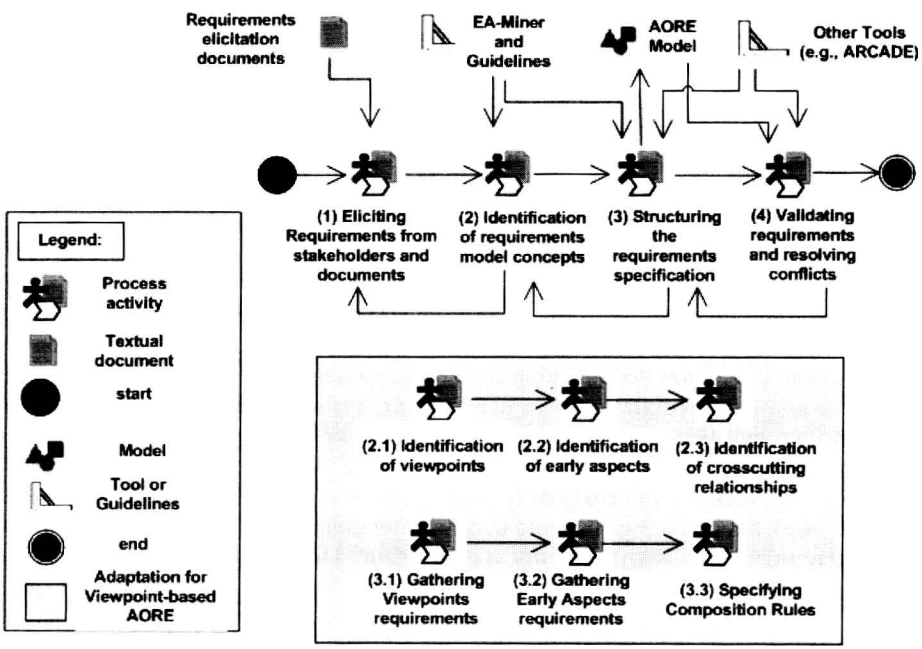

Fig. 1. General AORE process with detailed adaptation for viewpoint-based AORE

[1] Early aspects are also called concerns in this approach.