Ana Moreira · Ruzanna Chitchyan
João Araújo · Awais Rashid *Editors*

Aspect-Oriented Requirements Engineering

Springer

Ana Moreira · Ruzanna Chitchyan
João Araújo · Awais Rashid *Editors*

Aspect-Oriented Requirements Engineering

Springer

Aspect-Oriented Requirements Engineering

Ana Moreira • Ruzanna Chitchyan • João Araújo
Awais Rashid
Editors

Aspect-Oriented Requirements Engineering

Springer

Editors
Ana Moreira
João Araújo
Universidade Nova de Lisboa
Caparica, Portugal

Ruzanna Chitchyan
University of Leicester
Leicester, United Kingdom

Awais Rashid
University of Lancaster
Lancaster, United Kingdom

ISBN 978-3-642-38639-8 ISBN 978-3-642-38640-4 (eBook)
DOI 10.1007/978-3-642-38640-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013952959

ACM Computing Classification (1998): D.2, K.6

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Introduction

Aspect-oriented requirements-engineering (AORE) approaches aim to facilitate identification and analysis of *crosscutting concerns* (also termed as *aspects*) during requirements engineering to understand their potential effects and trade-offs with respect to other stakeholder requirements.

Often AORE approaches extend existing requirements-engineering techniques with additional support for identification, modularisation, composition, and analysis of crosscutting concerns. Such support is missing in most contemporary requirements-engineering techniques. For instance, in the classical use cases approach [1], non-functional requirements (NFR) cannot be readily modelled. Although techniques such as goal-based approaches [2, 3] support modularisation and analysis of such NFRs, they lack effective composition mechanisms to reflect and explore the complex dependencies and interactions (between NFRs themselves as well as NFRs and functional concerns) fully. Thus, AORE focuses on providing systematic means for modularisation, composition, and analysis of crosscutting concerns in requirements.

In the recent years significant work has been carried out in aspect-oriented requirements engineering. The aim of this book is to serve as a consolidation medium. The message given here is that whatever requirements engineering approach one uses, there will be a problem of treatment of broadly scoped concerns, which repeatedly appear, often have system-wide effects, and interact (e.g. conflict or supplement) with other requirements as well as influence the architectural decisions for the system-to-be. In this book we discuss how such *aspects* can be identified, represented, composed, and reasoned about, as well as used in specific domains and in industry. Thus, the book does not aim to present or promote a particular aspect-oriented requirements engineering approach but aims to provide an understanding of the aspect-oriented perspective on requirements engineering: what challenges does it tackle that supplement the more established requirements

engineering work, what tasks and processes does it use, and how does it benefit its adopting community.

Use of the Crisis Management [4] case study has been advised throughout the book as the common medium for demonstration of the work presented in each chapter. This is to shelter the reader from having to understand a potentially large set of different examples and instead focus on the essence of each presented approach. However, in some chapters, where the application of AORE to a specific domain is of significance by itself (e.g. in chapters discussing use of AORE in industrial setting), the common case study has been omitted.

1 Getting Started: AORE Main Concepts

In AORE a *concern* is defined as a unit encapsulating (one or more) requirements related to a certain matter of interest. For instance, a use case or a viewpoint with its requirements is an example of a concern.

An *aspect* (or *crosscutting concern*) is a modularisation unit for those requirements that do not align well into the established single-type decomposition modularisation units. For example, while use case units are ideal for functionality modularisation, non-functional requirements do not fit into the use case structure, but normally crosscut several use cases. Therefore, an aspect at the requirements level is a broadly scoped property (represented by a single requirement or a coherent set of requirements), which affects multiple other requirements in the system so that it may constrain the specified behaviour of the affected requirements or influence the affected requirements to alter their specified behaviour.

As illustrative example, consider a security requirement that constrains a requirement providing access to certain types of data in the system so that only a certified set of users may access that data. Similarly, another security requirement may influence communication requirements by altering their behaviour to impose encryption constraints. The requirements affected by a requirements-level aspect may already have been partitioned using abstractions such as viewpoints, use cases, and themes. Figure 1 shows a requirements-level aspect affecting multiple requirements in such a partitioning. Composition specification is used to relate requirements-level aspects with the non-crosscutting requirements.

Some awareness of the crosscutting concerns existed in the Requirements Engineering community before AORE, for instance, in works on NFR/softgoals [5] and viewpoints [6]. However, this was a segmented perspective, with crosscutting concerns considered as a “special” type of concerns, with “unusual” properties. AORE, on the other hand, provides a general unified framework explaining the properties of the crosscutting requirements as the natural result of (the traditional) modelling of the multi-perspective world with a single type of modularisation unit (such as use cases and goals). Also, AORE underlines the need for composition of concerns and aims to provide an extensive support for it. Compositions are used for understanding and analysis of concern interdependencies—for detection of potential

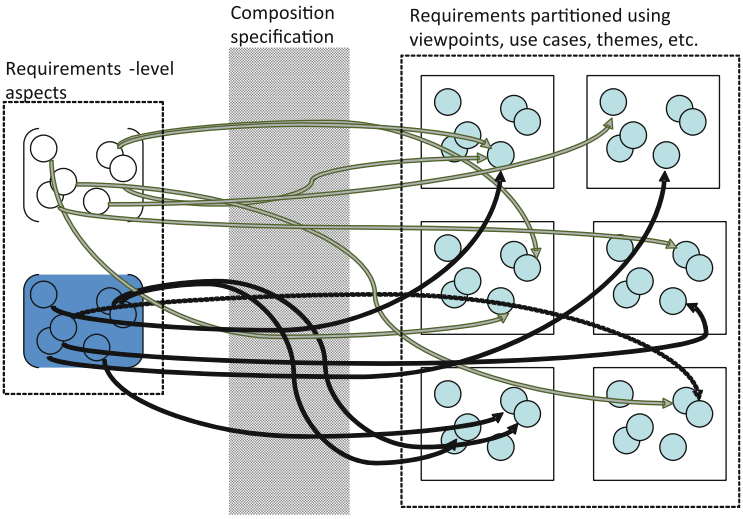

Fig. 1 Requirements-level aspects constraining and influencing (via a composition specification) other requirements

conflicts between various concerns/requirements very early on in order to either take corrective measures or make appropriate decisions for the next development step. The composed requirements also become valuable sources of validation for the complete system, as well as potential artefacts for requirement reuse.

Therefore, if requirements aspects are not effectively modularised, it is not possible to reason about their effect on the system or on each other, and the lack of modularisation of such properties can result in a large ripple effect on other requirements or architectural components upon evolution. The provision of effective means for handling aspects in requirements makes it possible to establish critical trade-offs early on in the software lifecycle.

Figure 2 depicts a general AORE framework, highlighting in grey the activities where AORE makes its major contribution.

One other issue noted by the AORE work is the need to trace crosscutting properties across the lifecycle of a software system. It is not sufficient to identify and reason about crosscutting concerns during requirements engineering. Once these concerns and their associated trade-offs have been established, it is essential that the software engineers can trace them to architecture (illustrated in Fig. 2 by the concern mapping activity), design, implementation and subsequent maintenance and evolution. Modularisation of crosscutting properties at the requirements level is the first step towards establishing such traceability.

As in other AO software lifecycle stages, AORE uses the concepts of joinpoints, pointcuts, advice, intertype declarations, and composition (or weaving) [7]. These concepts are normally interpreted somewhat differently for each individual AORE approach and the full or partial set of these concepts may be utilised by each given

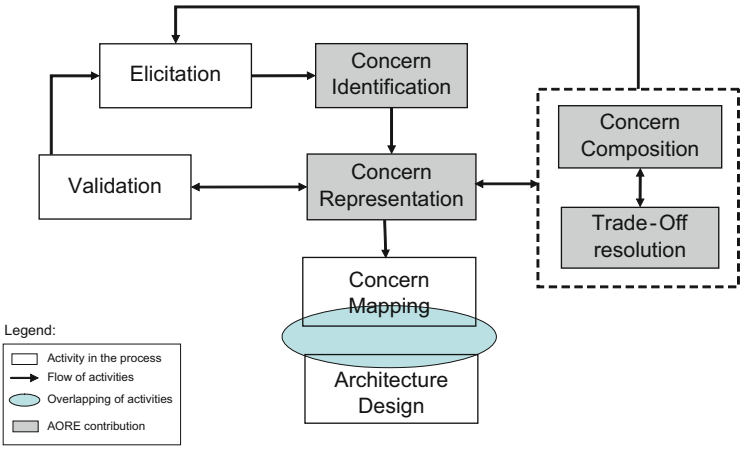

Fig. 2 AORE in the broader context of Requirements Engineering

work. However, presence of (any of) these notions in an RE approach will generally indicate presence of an AO perspective in it. These notions are generically defined below:

- Aspects can only be invoked, or composed with other modules, at some well-defined and principled points within the software artefacts. These points are referred to as *joinpoints*. As stated in [8]: "A joinpoint is a point of interest in some artifact in the software lifecycle through which two or more concerns may be composed. A *joinpoint model* defines the kinds of joinpoints available and how they are accessed and used". Examples of such well-defined, principled points in AORE are, for instance, goal notes, or tasks in goal graphs, or identifiable (e.g. via ID or name) requirements and concerns encapsulating those requirements, identifiable (e.g. via their grammatical role or meaning) parts of text, etc.
- *Pointcuts* specify a set of joinpoints at which a given aspect should interact with some other modules. Pointcuts can be defined *by extension*, i.e. by enumerating each joinpoint relevant for the given aspect application, or *by intension*, i.e. via a more abstract selection criteria, such as regular expressions, or semantic-matching queries. Because aspects normally broadly affect a number of other concerns, defining their interactions by extension is rather inefficient. Consequently, in AOSD, pointcuts are normally defined by intension [7]. Thus, a *pointcut* normally is a predicate that matches joinpoints.
- As noted above, an aspect affects a set of other concerns at the joinpoints. In AOSD terminology, it is said that aspects *advise* other concerns. An *advice* represents the particular part of aspect that will manifest itself (e.g. by adding or changing behaviour) at a given joinpoint of the affected concern. Traditionally, an advice in AOSD implies a *behaviour-related interaction* between aspectual

and non-aspectual artefacts. Such an interaction is also defined in respect with some temporal, conditional, or unconditional order.

- *Intertype declarations* (also called *introductions*) are an additional mechanism for directly modifying the *structure* of the original artefacts. For instance, an intertype declaration may insert a new requirement into a viewpoint or even change subtype structure, etc.
- *Composition* (also called *weaving*) is the integration of the separated crosscutting elements back into the modules crosscut by them. However, in AORE, it is not always necessary to physically integrate the aspectual elements into other modules [7]. Often a *composition specification* is sufficient for reasoning about aspectual and non-aspectual module interactions. Thus, composition here can often imply projecting the constraints and influences of individual requirements-level aspects on other system requirements, based on the knowledge inherent in the composition specification.

The composition specification define which aspectual elements (advice, intertype declarations, and so forth.) affect which joinpoints (selected by pointcuts) of which other modules, in what way, and defines what are the temporal, conditional, or unconditional circumstances of aspect invocation.

In summary, AORE uses the above outlined concepts to provide improved separation of concerns and composition at the requirements level. The composition definitions are often used as an analysis tool for conflict-point identification and interaction understanding, and, in some cases [9], transformational compositions are also realised.

It is also essential to note that not all aspectual artefacts identified at the requirements level will subsequently be represented as code-level aspects. On the contrary, some may well transform into other software artefacts (e.g. architectural topology) or business-related decisions (e.g. procedures for security policy used by the business) before an application is implemented. In addition, new aspects, often related to the selected development technology, will emerge at the other stages of software development, but these will not be visible in requirements.

2 Structure of the Book

This book is largely alighted with the main AORE-related activities depicted in Fig. 2: concern identification-related issues are discussed in the similarly titled Part I; topics on concern representation and composition are discussed in Part II titled Concern Modelling and Composition; topics of concern mapping (e.g. architectural implications of requirements level aspects and aspects in particular domains) are presented in Part III titled Domain-Specific Use of AORE; the issues of trade-off, conflicts, and validity are discussed in Part IV, under the title of Interaction Analysis; finally, under the title AORE Evaluation Part V presents two perspectives

on how AORE is used in industry and an overview chapter on evaluation work in AORE so far.

The **Concern Identification** section discusses crosscutting concern identification in textual as well as model-based requirements. The aim of concern identification is to, first of all, facilitate building the knowledge on what crosscutting occurs in requirements and why. Along with such knowledge collection, identification should be accompanied with modularisation support, which, ideally, can propagate the modularity to later stages of software development. In this section, Chap. 1 describes the EA-Miner tool-based approach, which offers automated support for identifying crosscutting in such requirements artefacts as viewpoints or use cases, which consist of natural language text. The main characteristic of this approach is the use of natural language processing (NLP) for concern identification. Chapter 2 presents a goal-based approach that uses a list of adaptation rules for the requirements aspects to be managed at runtime. It explains how different concepts in requirements aspects are formulated and reasoned about. The basic adaptation rules are classified according to the roles played in the runtime changes.

The **Concern Modelling and Composition** section, which includes Chaps. 3–7, is focused on modelling and composition definition in AORE. Since AORE defines an aspect as a new type of module with its particular rules of interacting with other modules, it is essential to deliver a good modelling support for representation of these new modules and their interrelationships with each other as well as non-aspectual modules. There are the challenges tackled in the present section.

Chapter 3 introduces an aspectual scenario-based approach where sequence diagram and state machines are modelled using a technique for modelling and composition of patterns based on graph transformations called MATA (Modelling Aspects Using a Transformation Approach).

Chapter 4 describes a semantics-based composition approach applied to textual requirements. Here the composition specifications are based on the semantics of the natural language. This is achieved by annotating the natural language requirements with information on their grammatical and semantic properties, and using these annotations as well as natural language semantics as a joinpoint model for composition specification.

Chapter 5 presents the composition mechanism for aspect-oriented user requirements notations (AoURN). The focus is on interleaving and enhanced matching based on semantics composition rules. Interleaved composition allows two scenarios to be combined keeping the overall behaviour of the original scenarios. Semantics-based matching allows for a class of refactoring operations to be performed on an AoURN model without breaking the matches of an aspect's pattern.

Chapter 6 presents AOV-graph, an approach that deals with the crosscutting problems arising from interactions in goal models. This approach helps in defining a crosscutting relationship which modularises interactions and provides composition and visualisation mechanisms to analyse and model the goal-based requirements.

Chapter 7 shows how to identify and model crosscutting concerns in Problem Frames. This is particularly relevant as in problem frames some requirements appear in several (sub) problems diagrams, resulting in scattering effect. This work

shows how to compose such concerns with the elements they crosscut via a textual composition language.

The **Domain-specific use of AORE** section discusses specific uses of requirements-level aspects, like architecture derivation from requirements, modelling security requirements with aspects, and volatile requirements modelling. This section demonstrates how effectively AORE can be used in more specific contexts. This part consists of Chaps. 8–10.

Chapter 8 offers a strategy to derive architectural component-based model from an aspect-oriented requirements specification. It uses model-driven development where meta-models and transformations are specified and implemented.

Chapter 9 presents an approach for handling changes made to security-critical programmes. The authors observe that when a change happens (in any part of a system), the validation procedure for the security requirements may need to be updated even if the security requirements have not changed.

Chapter 10 demonstrates how can help to deal with volatile (i.e. highly unstable) requirements. Here it is noted, that although volatile concerns are not always crosscutting, they have the same issues of independency, modular representation and composition that are required for aspects. Thus, AO perspective is particularly fruitful in this context. Moreover, the chapter discusses how evolution, constrained by volatile requirements, is facilitated via adoption of an aspect-oriented approach.

Aspects bring with themselves a new set of challenges of handling independencies and interactions. These challenges are discussed in the **Interaction Analysis** section. Aspects composition may result in undesirable behaviour that violates the overall systems requirements. These interactions happen due to side effects introduced by aspect composition, such as interference or negative contributions. These are discussed in Chaps. 11–14.

Chapter 11 shows an approach and tool called EA-Analyzer that automates the process of detecting conflicts within textual AO requirements specifications. The aim is to facilitate the requirements engineers' work with large natural language specifications, which may contain numerous interdependencies. An empirical evaluation of the tool is also discussed, showing that conflicts within AO textual requirements specifications can be detected with a good accuracy.

Chapter 12 presents a tool-supported approach for conflict management at the AORE level. It uses a hybrid multi-criteria analysis technique to perform trade-offs analysis and obtain a ranking of concerns. This technique can be used to support architectural choices during the software architecture design and "what-if" scenario analysis.

Chapter 13 presents a use case-driven approach and tool for analysing consistency at the level of requirements modelling. Activities are used to refine use cases and are combined with a specification of pre-and post-conditions into an integrated behaviour model. This is formalised using the graph transformation theory and used for reason about consistency.

Chapter 14 shows an approach where features are treated as aspects and feature composition as aspect composition. They use Composition Frames to compose aspects and resolve aspect interactions at runtime.

The **AORE evaluation** section describes experiences of use of AORE in industry and presents an overview of AORE evaluation work so far. The first part consists of Chaps. 15 and 16. Chapter 15 discusses how the technique called Requirements Composition Table (RCT) is used in two financial applications. The RCT technique has been implemented for a number of Wall Street applications at various investment banks. This chapter illustrates how RCT can help perform change impact analysis for releases and assess test coverage of existing regression test suites.

Chapter 16 discusses the application and evaluation of two AORE approaches (Theme/Doc and MDSOCRE) in the slot machines domain. This application involved several large requirement documents that have ambiguity issues and aspectual interactions.

Finally, Chap. 17, which also concludes the book, draws upon experience from evaluation performed in other phases of development and also the problems that can be experienced when evaluating AORE approaches to establish a series of guidelines to assist software developers.

3 Crisis Management System Case Study

In order to observe how the same problem is addressed by different approaches, we adopt a common case study to be used throughout the chapters of this book. The case study domain is crisis management systems, i.e. systems that manage the relevant parties, activities and information involved in solving a crisis. This case study was proposed as an exemplar to evaluate aspect-oriented modelling approaches in 2010 [4] and has since been used by the AOSD community for the evaluation and comparison of aspect-oriented approaches (e.g. CMA workshop series). The requirements used to create this exemplar were based on the real requirements document for crisis management systems created by Optimal Security [10].

The general objectives of a Crisis Management System (CMS) is to assist in the coordination of a crisis; to guarantee that a catastrophic situation is under control; to mitigate the crisis effects by allocating and managing the available resources in an effective manner; to identify, create, and execute missions in order to manage the crisis; and to recover the crisis information to allow future analysis [4]. A crisis can range from natural disasters (e.g. earthquakes, fire, floods), terrorist attacks or sabotage (e.g. explosions, kidnapping), accidents (e.g. car crash plant explosion, plane crash) and technological disruptions. All these are examples of emergency situations that are unpredictable and can lead to severe after-effects unless handled immediately.

A crisis management system facilitates the communication and interoperation between all stakeholders involved in handling the crisis (e.g. government, police systems, medical services, military systems). A CMS allocates and manages resources, and facilitates access to relevant information to authorised users of the CMS.

Finally, several non-functional properties are related to CMS, e.g. availability, response time, security, safety, mobility, persistence and multi-access. They are broadly scope and potentially crosscutting. The full case study documents are provided in the Appendix A of this book.

4 Intended Audience

This book is intended for software developers, software engineers, industrial trainees, and undergraduate and postgraduate students.

5 Acknowledgements

We are grateful to many people and several institutions for their contributions on the development of AORE.

To our co-organisers of Early Aspects workshop series. Sixteen editions have been organised since 2002. Thank you to Paul Clements, Bedir Tekinerdogan, and Elisa Baniassad for helping with the steering of the workshop, and to Alberto Sardinha, Alessandro Garcia, Carla Silva, Christa Schwanninger, Gunter Mussbacher, Jan Gerben Wijnstra, Jeff Gray, Jon Whittle, John Grundy, Mónica Pinto, Nan Niu, Pablo Sanchez, Paulo Merson, Uirá Kulesza, and Vander Alves, for taking the lead of the organisation in different editions. We are grateful for the interest demonstrated by all the authors who submitted their work and helped so much in creating the Early Aspects community, to the Program Committee members for offering their time to review the papers, to the participants that helped keeping discussions alive, to everyone that contributed to the increase of the number of postgraduate students that accomplished their dissertations in the field of Early Aspects, and to all the reviewers of the chapters included in this book.

A special word of thanks goes to Pete Sawyer for the discussions we had with him from the time the idea born in our minds. He contributed with the first vision paper on AORE, and his expertise and extensive experience on Requirements Engineering helped us finding an integrated view for RE with aspects.

During these 15 years of work, several people contributed to the development of Early Aspects through several projects funded by European Union, EPSRC, Fundação para a Ciência e Tecnologia (FCT), Conselho de Reitores das Universidades Portuguesas (CRUP) (bilateral projects with France and Spain), and CAPES/GRICES (bilateral projects between Brazil and Portugal). A special thanks to the European AMPLE and AOSD-Europe projects.

Finally, we thank to CITI research centre at Universidade de Nova Lisboa for funding several visits of Awais to Lisbon.