Early Aspects: a Model for Aspect-Oriented Requirements Engineering

Awais Rashid[†], Peter Sawyer[†], Ana Moreira[‡], João Araújo[‡]
[†]*Computing Department, Lancaster University, Lancaster LA1 4YR, UK*
[‡]*Dept. Informática, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal*
{awais, sawyer}@comp.lancs.ac.uk, {amm, ja}@di.fct.unl.pt

Abstract

Effective RE must reconcile the need to achieve separation of concerns with the need to satisfy broadly scoped requirements and constraints. Techniques such as use cases and viewpoints help achieve separation of stakeholders' concerns but ensuring their consistency with global requirements and constraints is largely unsupported. In this paper we build on recent work that has emerged from the aspect-oriented programming (AOP) community to propose a general model for aspect oriented requirements engineering (AORE). The model supports separation of crosscutting functional and non-functional properties at the requirements level. We argue that early separation of such crosscutting properties supports effective determination of their mapping and influence on artefacts at later development stages. A realisation of the model based on a case study of a toll collection system is presented.

1. Introduction

The *separation of concerns* principle [4] proposes encapsulating features into separate entities in order to localise changes to them and deal with one important issue at a time. For example, the UML uses different models to deal with different properties of a problem domain separately. In RE, viewpoints [6] have been advocated as a means of partitioning requirements as a set of partial specifications that aid traceability and consistency management.

The focus of this paper is on concerns that cut across other concerns. These *crosscutting* concerns are responsible for producing tangled representations that are difficult to understand and maintain. Examples of such concerns at the implementation level are distribution and synchronisation code that cannot be encapsulated in one class and is typically spread across several classes. Aspect-oriented software development [5] aims to identify and specify such crosscutting concerns in separate modules, known as *aspects*, so that localisation can be promoted. This results in better support for modularisation hence reducing development, maintenance and evolution costs.

A number of aspect-oriented programming (AOP) approaches have been proposed. These range from linguistic mechanisms [1] to filter-based techniques [5] through to traversal-oriented [5] and multi-dimensional approaches [11] [13]. Work has also been carried out to incorporate aspects, and hence separation of crosscutting concerns, at the design level mainly through extensions to the UML meta-model e.g. [3]. Research on the use of aspects at the requirements engineering stage is still immature and there is no consensus about what an aspect is at this early stage of software development and how it maps to artefacts at later development stages.

An aspect-oriented requirements engineering approach targeted to component based software development has been proposed in [7]. There is a characterisation of diverse aspects of a system that each component provides to end users or other components. However, the identification of aspects for each component is not clearly defined. Separation of crosscutting properties has also been considered in [12] which proposes a viewpoint-oriented requirements engineering method called PREView. A PREView viewpoint encapsulates partial information about the system. Requirements are organised in terms of several viewpoints, and analysis is conducted against a set of concerns intended to correspond broadly to the overall system goals. Due to this broad scope concerns crosscut the requirements emerging from viewpoints. In applications of the method, the concerns that are identified are typically high-level non-functional requirements. Beyond alerting the requirements engineer to the risk that viewpoint requirements and concerns may cause inconsistencies, the approach does not identify the mapping or influence of crosscutting properties on artefacts at later development stages.

The above discussion highlights the need to include aspects as fundamental modelling primitives at the requirements engineering level. The motivations for this are two-fold:

1. Providing improved support for separation of crosscutting functional and non-functional properties during requirements engineering hence offering a better means to identify and manage conflicts arising due to tangled representations;
2. Identifying the mapping and influence of requirements level aspects on artefacts at later

development stages hence establishing critical trade-offs before the architecture is derived.

This paper proposes a model for aspect-oriented requirements engineering aimed as a stepping-stone towards the above goals. The model, discussed in section 2, supports effective determination of the mapping and influence of aspects on later development stages. Section 3 describes the application of the model to a case study of a toll collection system. Section 4 concludes the paper by discussing key outstanding issues and directions for future work.

2. A model for AORE

Modern systems have to run in highly volatile environments where the business rules change rapidly. Therefore, systems must be easy to adapt and evolve. If not handled properly, crosscutting concerns inhibit adaptability. It is therefore essential to think about crosscutting concerns as early as possible. The model we envisage to deal with crosscutting concerns at the requirements level is composed of six activities (cf. fig. 1). This is based on treating PREView concerns as adaptations of the AOP notion of aspects.

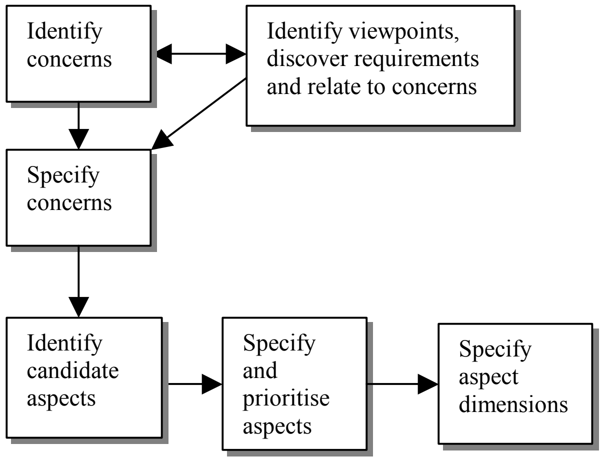
Identify concerns
Identify viewpoints, discover requirements and relate to concerns
Specify concerns
Identify candidate aspects
Specify and prioritise aspects
Specify aspect dimensions

Figure 1. AORE model

To begin with, we need to identify concerns and discover requirements. The order in which these two activities are accomplished is dependant on the dynamics of the interaction between requirements engineers and the stakeholders. In any case, it is useful to relate concerns to requirements as the former may constrain the latter. The next activity is to specify concerns in more detail. If a concern crosscuts several requirements (i.e. if a concern may influence or constrain more than one viewpoint) it is considered a *candidate aspect*. This is followed by a detailed specification of candidate aspects. This can lead us to refine aspects, making them more concrete, and to identify interactions and conflicts between them [9]. In order to resolve conflicts among aspects a prioritisation approach is used. The last activity in the model is identification of the dimensions of an aspect. We have observed that aspects at this early stage can have an impact that can be described in terms of two dimensions:

- *Mapping*: an aspect might map onto a system feature/function (e.g. a simple method), decision (e.g. a decision for architecture choice) and design (and hence implementation) aspect (e.g. response time). This is the reason we have chosen to call aspects at the RE stage *candidate aspects* as, despite their crosscutting nature at this stage, they might not directly map onto an aspect at later stages.
- *Influence*: an aspect might influence different points in a development cycle, e.g. availability influences the system architecture while response time influences both architecture and detailed design.

Note that prioritisation of aspects should precede the identification of their dimensions as conflict resolution may be required when determining *influence*.

3. Applying the model to a case study

The case study is a simplified version of the toll collection system on the Portuguese highways [2]:

"In a road traffic pricing system, drivers of authorised vehicles are charged at toll gates automatically. The gates are placed at special lanes called green lanes. A driver has to install a device (a gizmo) in his/her vehicle. The registration of authorised vehicles includes the owner's personal data, bank account number and vehicle details. The gizmo is sent to the client to be activated using an ATM[1] that informs the system upon gizmo activation.

A gizmo is read by the toll gate sensors. The information read is stored by the system and used to debit the respective account.

When an authorised vehicle passes through a green lane, a green light is turned on, and the amount being debited is displayed. If an unauthorised vehicle passes through it, a yellow light is turned on and a camera takes a photo of the plate (used to fine the owner of the vehicle). There are three types of toll gates: single toll, where the same type of vehicles pay a fixed amount, entry toll to enter a motorway and exit toll to leave it. The amount paid on motorways depends on the type of the vehicle and the distance traveled."

To help identify the crosscutting concerns we have used the requirements elicitation tool JPREView[2]. In this approach, viewpoints are specified through a template. A viewpoint template consists of a viewpoint name, focus,

[1] Portuguese ATMs offer a wide range of services, e.g. selling train or theatre tickets.
[2] http://www.comp.lancs.ac.uk/computing/research/cseg/projects/deada/JPreview.html

an optional list of sub-viewpoints, a list of concerns, and a list of requirements. Concerns are elaborated by deriving a set of requirements which we call *external* requirements. Our approach is not limited to viewpoints. We can use: goal-oriented requirements which cover functional and non-functional concerns [10]; use cases or scenario-based approaches, by specifying which use cases/scenarios are crosscut by a concern; problem frames which can be viewed as concerns [8].

3.1. Identify concerns

Concerns are identified by analysing the initial requirements. For example, since the owner of a vehicle has to indicate his/her bank details during registration, *security* is an issue that the system needs to address. Other concerns in our case study, identified in a similar fashion, are: *Response Time, Multi-user System, Compatibility, Legal Issues, Correctness* and *Availability*.

3.2. Specify concerns

For simplification we choose to specify only two concerns here: Compatibility and Response Time. The choice is aimed at demonstrating a range of dimensions (cf. section 3.6).

Concern: Compatibility
External Requirements:
1. Users will activate the gizmo using an ATM.
2. The police will deal with vehicles using the system without a gizmo.

Concern: Response-Time
External Requirements:
1. A toll gate has to react *in-time* in order to:
   1.1 read the gizmo identifier;
   1.2 turn on the light (to green or yellow) before the driver leaves the toll gate area;
   1.3 display the amount to be paid before the driver leaves the toll gate area;
   1.4 photograph the unauthorised vehicle's plate number from the rear.
2. The system needs to react *in-time* when:
   2.1 The user activates the gizmo using an ATM.

3.3. Discover requirements and relate to concerns

The following viewpoints were identified: *ATM, Vehicle, Gizmo, Police, Debiting System,Entry Toll, Exit Toll, Driver, Vehicle-Owner* and *System Administrator*.

Having identified concerns and viewpoints, we must relate them. We do this using the templates of JPREView. In this example, we have chosen the viewpoints *ATM and Exit Toll*. Viewpoint focus has been omitted for simplification. An ATM allows customers to enter their own transactions using cards. The ATM sends the transaction information for validation and processing.

Viewpoint: ATM.
Concerns: Security, Compatibility, Response time
Requirements:
1. The ATM will send the customer's card number, account number and gizmo identifier to the system for activation.
2. The ATM will send the account number to the system to obtain the gizmo identifiers associated with the account.
3. The ATM will send the account number, new card number and the gizmo identifier to the system to update the card number and reactivate the gizmo.

Viewpoint: Exit Toll
Concerns: Response Time, Correctness, Legal Issues
Requirements:
1. The driver will see a yellow light if s/he did not use an entry toll.
2. The amount being debited depends upon the entry point.

3.4. Identify candidate aspects

The requirements number 1 and 2 of the Response Time concern crosscut the requirements of two different viewpoints (ATM and Exit Toll). Consequently the Response Time concern qualifies as a candidate aspect. In fact, in our case study all the concerns identified form candidate aspects as they cut across multiple viewpoints. However, in another system a concern might constrain a single viewpoint and, hence, will not qualify as a candidate aspect.

3.5. Specify and prioritise aspects

Specification provides an opportunity to refine the aspects and make them more concrete, e.g. "Legal Issues" can be refined to "Legal Issues for Unauthorised Vehicles", "Legal Issues for Billing", etc. Prioritisation allows us to describe the extent to which an aspect may constrain a viewpoint e.g. security is a concern for the "Vehicle Owner" viewpoint. Security must be assigned a high priority during registration as the vehicle owner provides personal information such as bank account details. However, security is not such a high priority for correspondence as to use a personal courier.

3.6. Specify aspect dimensions

Specification of a candidate aspect's dimensions makes it possible to determine its influence on later development stages and identify its mapping onto a function, decision or aspect. Consider our Compatibility candidate aspect. The requirements derived from this aspect will influence parts of the system specification, architecture and design pertaining to requirements derived from viewpoints constrained by it. They will also influence system evolution as change of the user's ATM cards must be anticipated. The Compatibility aspect will, however, map on to a function allowing activation and reactivation of the gizmo. The Response Time concern, on the other hand, will influence the type of architecture chosen and the design of the classes realising the requirements constrained by Response Time. It will map to an aspect at the design and implementation level because response time properties cannot be encapsulated in a single class and will be otherwise spread across a number of classes. The various candidate aspects in our case study and their mappings and influences are shown in Table 1.

Table 1: Aspect dimension specification

Candidate aspect	Influence	Mapping
Compatibility	Specification, architecture, design, evolution	Function
Response time	Architecture, design	Aspect
Legal issues	Specification	Function
Correctness	Specification, design	Function
Security	Architecture, design	Aspect
Availability	Architecture	Decision
Multi-user system	Architecture, design	Aspect

4. Conclusions

This paper has proposed a model for aspect-oriented requirements engineering. The model supports separation of crosscutting properties from early stages of the development and identification of their mapping and influence on later development stages. This makes it possible to identify conflicts and establish possible trade-offs early on in the development cycle and promotes traceability of broadly scoped requirements and constraints throughout system development, maintenance and evolution. The improved modularisation and traceability obtained through early separation of crosscutting concerns can play a central role in building systems resilient to unanticipated changes hence meeting the adaptability needs of volatile domains such as banking, telecommunications and e-commerce.

With increasing support for aspects at the design and implementation level, the inclusion of aspects as fundamental modelling primitives at the requirements level and identification of their mappings also helps to ensure homogeneity in an aspect-oriented software development process.

Our future work will focus on validation of aspects, their composition with other requirements and resolution of possible conflicts resulting from the composition process. We also aim to develop a notation to describe aspects, their interactions and composition relationships at the requirements level.

References

[1] Xerox PARC, USA, "AspectJ Home Page", http://aspectj.org/, 2002.

[2] R. Clark and A. Moreira, "Constructing Formal Specifications from Informal Requirements", Software Technology and Engineering Practice, 1997, IEEE Computer Society Press, pp. 68-75.

[3] S. Clarke and R. J. Walker, "Composition Patterns: An Approach to Designing Reusable Aspects", International Conference on Software Engineering (ICSE), 2001.

[4] E. W. Dijkstra, *A Discipline of Programming*. Englewood Cliffs, NJ: Prentice Hall, 1976.

[5] T. Elrad, R. Filman, and A. Bader (eds.), "Theme Section on Aspect-Oriented Programming", *Communications of ACM*, Vol. 44, No. 10, 2001.

[6] A. Finkelstein and I. Sommerville, "The Viewpoints FAQ." *BCS/IEE Software Engineering Journal*, Vol. 11, No. 1, 1996.

[7] J. Grundy, "Aspect-Oriented Requirements Engineering for Component-based Software Systems", 4th IEEE International Symposium on RE, 1999, IEEE Computer Society Press, pp. 84-91.

[8] M. Jackson, *Problem Frames: Analyzing and Structuring Software Development Problems*: Addison Wesley, 2000.

[9] H. Klaeren, E. Pulvermueller, A. Rashid, and A. Speck, "Aspect Composition Applying the Design by Contract Principle", 2nd International Symposium on Generative and Component-based Software Engineering (GCSE), 2000, Springer-Verlag, LNCS 2177, pp. 57-69.

[10] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", 5th International Symposium on Requirements Engineering, 2001, IEEE Computer Society Press, pp. 249-261.

[11] A. Rashid, "A Hybrid Approach to Separation of Concerns: The Story of SADES", Proc. Reflection conf. 2001, Springer-Verlag, LNCS, 2192, pp. 231-249.

[12] I. Sommerville and P. Sawyer, *Requirements Engineering - A Good Practice Guide*: John Wiley and Sons, 1997.

[13] P. L. Tarr, H. Ossher, W. H. Harrison, and S. M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns", International Conference on Software Engineering (ICSE), 1999, ACM, pp. 107-119.