Conflict Management in Aspect-Oriented Requirements Engineering[*]

Alberto Sardinha
Dept. de Eng. Informática
Instituto Superior Técnico, UTL
Lisboa, Portugal
sardinha@acm.org

João Araújo
Dept. de Informática, CITI/FCT
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
ja@di.fct.unl.pt

Ana Moreira
Dept. de Informática, CITI/FCT
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
amm@di.fct.unl.pt

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

Abstract
One of the main goals of Aspect-Oriented Requirements Engineering (AORE) is to address the composability and subsequent analysis of crosscutting requirements. However, composing requirements in AORE may lead to conflicting situations that have to be analyzed and resolved. There are a few AORE methods for resolving conflicts between requirements, but most of them are error-prone or have scalability issues. This paper presents a mathematical formulation for resolving conflicts that can be implemented with Search-based techniques (i.e., metaheuristics) to address the scalability and error-proneness of the existing AORE methods.

1. Introduction
Aspect-Oriented Requirements Engineering (AORE) [1, 3, 11] aims at addressing the composability and subsequent analysis of crosscutting requirements. Compositions in AORE are utilized to explicitly represent and analyze the interdependencies between requirements.

However, composing requirements may lead to situations where conflicting dependencies between requirements have to be detected and resolved. In this context, a conflict occurs when a crosscutting requirement has a negative contribution with another crosscutting requirement in the same base requirement. For example, a data encryption requirement and a response time requirement that crosscut the same base requirement may lead to a conflicting dependency, because encryption normally reduces the responsiveness of a system [13].

Several AORE approaches for resolving conflicts are either based on simple reasoning methods [10, 12] that are error-prone or Multiple Criteria Decision Making [4] models that cannot handle more than twenty requirements [2]. Therefore, the resolution of conflicts in large AORE specifications is an error-prone and time-consuming task that creates a burden on the requirements engineer.

In order to deal with the scalability and error-proneness issues of the existing AORE approaches, this paper presents a mathematical formulation for resolving conflicts in AORE specifications that can be implemented with search-based techniques. Within the Search-Based Software Engineering field [8], search-based techniques are metaheuristics that are utilized for solving software engineering problems that can be formulated as optimization problems.

This paper is organized as follows. Section 2 presents the mathematical formulation for resolving conflicts in AORE. Section 3 discusses how to implement the mathematical formulation with a search-based technique. Finally, the conclusions and future work are presented in Section 4.

2. Resolving Conflicts in AORE
In our multi-dimensional approach, a concern implies any coherent collection of requirements. All concerns are treated in a uniform fashion, so we do not classify concerns into viewpoints, use cases or aspects; However, our concerns still encapsulate coherent sets of functional and non-functional requirements. For example, a security con-

[*]An earlier version of this paper was presented at the Early Aspects 2010 workshop at the 9th International Conference on Aspect-Oriented Software Development. The paper is recommended by Ruzanna Chitchyan and Steffen Zschaler.

Sardinha, A., Araújo, J., Moreira, A., and Rashid, A. 2010. Conflict Management in Aspect-Oriented Requirements Engineering. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Early Aspects, R.Chitchyan, S. Zsachaler (Eds.), Vol. 2, No. 1 (2010) 56-59

cern may contain a data encryption requirement and a security check requirement. The composition rules [12, 5] in aspect-oriented specifications define the relationship between various groups of requirements. However, the process of defining these composition rules may lead to conflicting dependencies that have to be detected and resolved.

The aim of this section is to present the main characteristics of the conflict management problem in AORE and to propose a model that formulates the conflict resolution problem as an optimization problem. The following characteristics were considered in this mathematical formulation:

1. *Aspect-oriented Specification*: This specification is composed of requirements that are grouped into concerns, and composition rules that define the relationship between requirements.
2. *Stakeholders' priorities and importance*: The requirements of the system are collected from stakeholders, and each stakeholder might have different priorities for each requirement. Moreover, each stakeholder might have a different level of importance to the organization.
3. *Available resources*: Every software project has a limited amount of resources (e.g., software developers, tools, licenses, financial resources) to implement the requirements.

In a general sense, our approach has as its basis a mathematical formulation of an aspect-oriented specification, stakeholder's priorities and importance, and resources available for implementing the requirements. The aim of the conflict management problem is to select requirements that can maximize the stakeholders' satisfaction and respect the availability of resources.

2.1 Aspect-oriented Specification

In general, an aspect-oriented specification is composed of the following elements:

- **Requirements** $R = \{r_1, r_2, ..., r_{|R|}\}$: Requirements are collected from different stakeholders (e.g, end-users, developers, managers) and express different perspectives on the system;
- **Concerns** $C = \{C_1, C_2, ..., C_{|C|}\}$: Concerns encapsulate one or more requirements related to a certain matter of interest; Hence, a concern is a subset of R: $C_i \subseteq R$.
- **Compositions rules** $Cr = \{Cr_1, Cr_2, ..., Cr_{|Cr|}\}$: A composition rule, Cr_j, describes how a set of constraint requirements, $Cs_j \subseteq R$, crosscut a set of base requirements, $Bs_j \subseteq R$.

2.2 Stakeholders' Priorities and Importance

Stakeholder [15] is a person or a group that will, in some way, be affected by the system. They normally have different objectives and needs that can lead to conflicting situations. Additionally, stakeholders have different importance levels to the organization.This is formulated as follows:

- **Stakeholders**: $S = \{s_1, s_2, ..., s_{|S|}\}$
- **Stakeholders' Priorities**: $priority : S \times R \rightarrow \mathbb{R}$. Hence, we use weights to express stakeholders's priorities.
- **Stakeholders' Importance Levels**: $importance : S \rightarrow \mathbb{R}$. Hence, we also use weights to express the different importance levels of the stakeholders.

2.3 Available Resources

Resources are all the services, goods, and human resources available for implementing the requirements (e.g, developers, tools, testers, hardware, software). This is formulated as follows:

- **Resources** $Rs = \{Rs_1, Rs_2, ..., Rs_{|Rs|}\}$

2.4 Identifying Match Points

Match Points are used as a basis to detect and resolve conflicts among composed requirements, because they explicitly represent the interactions of a requirement with other requirements with reference to a base requirement. This Section describes the algorithm used to identify Match Points, which is also utilized in [14] to detect potential conflicts between requirements.

Recall that a composition rule, Cr_k, describes how a set of constraint requirements, $Cs_k \subseteq R$, crosscut a set of base requirements, $Bs_k \subseteq R$. Additionally, let Sc_i be the set of compositions where r_i is a base requirement. Thus, the Match Point of requirement r_i is defined by equation 1.

$$M_i = \bigcup_{k \in Sc_i} Cs_k \tag{1}$$

A Match Point is the union of all the constraint requirements that crosscut the same base requirement. For example, a composition may select a constraint requirement r_1 = "The system should use a security protocol when sending data over the internet" to crosscut the base requirements r_3 = "The login and password are sent to the server". Additionally, another composition may select a constraint requirement r_2 = "The response time must not exceed 5 seconds" to crosscut the base requirement r_3. Thus, the Match Point of r_3 is $\{r_1, r_2\}$. So for each base requirement in the specification, we can find a Match Point.

2.5 Maximizing the Stakeholders' Satisfaction

In a software project, the requirements engineer should always try to maximize the stakeholders' satisfaction, without violating the availability of resources. Hence, this can be formulated as follows:

For each Match Point M_i:

$$Maximize : \sum_{s \in S} \sum_{r \in M_i} importance(s) \times priority(s, r) \times x_i \tag{2}$$

Subject to:

$$\sum_{r \in R} effort(r) \times x_i \leq resources \tag{3}$$

Table 1: The Importance of Requirements and Stakeholders

Very Important	$]0.8, 1.0]$
Important	$]0.5, 0.8]$
Average	$]0.3, 0.5]$
Not So Important	$]0.1, 0.3]$
Do Not Care	$]0.0, 0.1]$

where x_i are decision variables that assume a value 1 when requirement r should be implemented and a value 0 when it should not.

Equation 2 expresses the stakeholder's satisfaction by considering the stakeholder's importance level and priorities, and Equation 3 expresses that the effort needed to implement the requirements must respect the availability of resources.

For example, the Match Point in Section 2.4 has two conflicting requirements $\{r_1, r_2\}$ and only one developer (i.e., a resource) that can implement only one of the requirements. These two requirements are conflicting, because a security protocol (such as encryption) normally reduces the responsiveness of a system (so the data might not be sent within 5 seconds).

Assuming these requirements have been collected from two stakeholders ($\{s_1, s_2\}$), and each stakeholder selects weights to each requirement using the scales in [12] (Table 1):

Stakeholder	Requirement r_1	Requirement r_2
s_1	*Very Imp.* (1.0)	*Average* (0.5)
s_2	*Important* (0.8)	*Very Imp.* (1.0)

In addition, the organization regards the stakeholder s_1 to be *Very Important* (1.0 - Table 1), while the stakeholder s_2 is *Not So Important* (0.3 - Table 1). Hence, we can find the optimal solution with Equation 2:

$$1.0 \times 1.0 \times x_1 + 1.0 \times 0.5 \times x_2 +$$

$$0.3 \times 0.8 \times x_1 + 0.3 \times 1.0 \times x_2$$

Recall that we can only implement one of the requirements, so the only feasible solutions are $\{x_1 = 1, x_2 = 0\}$ and $\{x_1 = 0, x_2 = 1\}$. Hence, the optimal solution is the one that selects r_1 to be implemented ($\{x_1 = 1, x_2 = 0\}$).

2.6 Resolving Conflicts in AORE

In order to resolve conflicts, we propose the following activities:

1. *Detect Conflicts*: We must first detect the match points that have conflicting requirements (i.e., requirements that have a negative contribution with another crosscutting requirement in the same base requirement). There are many tools, such as [14], that can help a requirements engineer perform this activity.

2. *Solve the Optimization Problem*: Equation 2 and Equation 3 can be used to select the requirements that maximize the stakeholders' satisfaction. Moreover, the requirements in each match point can also be sorted by the stakeholders' satisfaction to create a ranking of the requirements. This activity can be performed by a search-based technique.

3. *Negotiate with Stakeholders*: If a match point still has conflicting requirements, then negotiation among stakeholders is needed. The output of the previous activity can be used to help the stakeholders reason about the conflicts and agree on a resolution. For example, a stakeholder might agree to lower a weight that expresses his priorities.

Once all the conflicts are resolved, the specification is revised and recomposition carried out to identify any further conflicts.

3. Utilizing Search-Based Techniques to Resolve Conflicts in AORE

Genetic Algorithm (GA) [9] is a machine learning technique that has been successfully applied [6, 7] to Search-Based Software Engineering problems. This learning method is motivated by an analogy to biological evolution, by searching a space of candidate solutions in order to identify the best solution.

The implementation of a GA is based on a pool (called population) of candidate solutions (called chromosomes), and iteratively updates this population by mutating and recombining parts of the best currently known chromosomes. The best chromosome is defined as the candidate solution that maximizes a predefined numerical value (also known as fittness).

In GAs, the candidate solutions (chromosomes) are encoded as binary strings. To represent the n match points and m requirements in each match point of our mathematical formulation in Section 2, we can use a string with $n \times m$ bits. Addtionally, the fitness of the population is derived from Equation 2 and 3.

4. Conclusions

This paper presents a mathematical formulation for resolving conflicts in AORE that can be implemented with a popular search-based technique called Genetic Algorithm. The mathematical formulation is composed of an aspect-oriented specification, stakeholders' importance and priorities, and available resources for implementing the requirements. We also show how to maximize the stakeholders' satisfaction and how to use it to resolve conflicts.

Our future research work will focus on the implementation of a tool and an empirical evaluation of the tool. We also would like to test different objective functions (Equation 2) in order to capture different variations of the stakeholders' satisfaction.

5. Acknowledgements

This work has been supported by a Marie Curie Fellowship from the European Commission (Grant Agreement No. PIIF-GA-2008-221016) and partially funded by FCT MCTES.

References

[1] J. Araujo, J. Whittle, and D.-K. Kim. Modeling and composing scenario-based requirements with aspects. In *RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International*, pages 58–67, Washington, DC, USA, 2004. IEEE Computer Society.

[2] P. Avesani, C. Bazzanella, A. Perini, and A. Susi. Facing scalability issues in requirements prioritization with machine learning techniques. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 297–306, Washington, DC, USA, 2005. IEEE Computer Society.

[3] E. Baniassad and S. Clarke. Theme: An approach for aspect-oriented analysis and design. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 158–167, Washington, DC, USA, 2004. IEEE Computer Society.

[4] I. S. Brito, F. Vieira, A. Moreira, and R. A. Ribeiro. Handling conflicts in aspectual requirements compositions. *T. Aspect-Oriented Software Development*, 3:144–166, 2007.

[5] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In *AOSD '07: Proceedings of the 6th international conference on Aspect-oriented software development*, pages 36–48, New York, NY, USA, 2007. ACM.

[6] F. Colares, J. Souza, R. Carmo, C. Padua, and G. Mateus. A new approach to the sotware release planning. In *Proceedings of the 2009 XXIII Brazilian Symposium on Software Engineering*, 2009.

[7] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46:243–253, 2004.

[8] M. Harman and B. F. Jones. Search-based software engineering. *Information & Software Technology*, 43(14):833–839, 2001.

[9] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[10] A. Moreira, A. Rashid, and J. Araújo. Multi-dimensional separation of concerns in requirements engineering. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005.

[11] A. Rashid, A. Moreira, and J. Araújo. Modularisation and composition of aspectual requirements. In *AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development*, pages 11–20, New York, NY, USA, 2003. ACM.

[12] A. Rashid, A. Moreira, and J. Araújo. Modularisation and composition of aspectual requirements. In *AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development*, pages 11–20, New York, NY, USA, 2003. ACM.

[13] A. Sampaio, P. Greenwood, A. F. Garcia, and A. Rashid. A comparative study of aspect-oriented requirements engineering approaches. In *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pages 166–175, Washington, DC, USA, 2007. IEEE Computer Society.

[14] A. Sardinha, R. Chitchyan, N. Weston, P. Greenwood, and A. Rashid. Ea-analyzer: Automating conflict detection in aspect-oriented requirements. In *ASE '09: Proceedings of the Twenty-Fourth IEEE/ACM International Conference on Automated Software Engineering*, 2009.

[15] I. Sommerville. *Software Engineering 8*. Addison-Wesley, eighth edition, 2007.