Conflict Identification with EA-Analyzer

Alberto Sardinha[1], Ruzanna Chitchyan[2], João Araújo[3], Ana Moreira[3] and Awais Rashid[4]

Abstract Conflict Identification in Aspect-Oriented Requirements Engineering (AORE) is an integral step toward resolving conflicting dependencies between requirements at an early stage of the software development. However, to date there has been no work supporting detection of conflicts in a large set of textual requirements without converting texts into an alternative representation (such as models or formal specification) or direct stakeholder involvement. Here, we present EA-Analyzer, an automated tool for identifying conflicts directly in aspect-oriented requirements specified in natural-language text. This chapter is centered on a case-study based discussion of the accuracy of the tool. EA-Analyzer is applied to the Crisis Management System, a case study used as an established benchmark in several areas of aspect-oriented research.

1 Introduction to Conflict Identification in Aspect-Oriented Requirements

Aspect-Oriented Requirements Engineering (AORE) [Rashid et al., 2003] aims at addressing the identification, representation, modularization, composition and subsequent analysis of crosscutting requirements. Identification and resolution of conflicts between concerns is often an essential part of the analysis activity. Since in AORE most concern inter-relationships can be defined via compositions, composition specifications are also a natural focus for conflict identification work. Thus, it is not surprising that a number of studies have been focusing on conflict detection and resolution via composition definitions [Brito et al., 2007; Sardinha et al., 2013; Weston et al., 2008].

As discussed in previous chapters, although most requirements documents tend to be written in natural language, research on conflict detection in aspect-oriented (AO) textual requirements tends to re-format the textual artifacts before starting the conflict identification process. For instance, some researchers tackle the

[1] INESC-ID and Instituto Superior Técnico, UTL, Portugal

[2] Department of Computer Science, University of Leicester, UK

[3] Informatics Department, CITI/FCT, Universidade Nova de Lisboa, Portugal

[4] Computing Department, Lancaster University, UK

conflict identification by first carrying out formalization of requirements and compositions [Laney et al., 2004; Mostefaoui and Vachon, 2007; Weston et al., 2008]; others represent requirements and compositions via models, then undertake model-based analysis [Mehner et al., 2006; Barais et al., 2008]; and, finally, others involve stakeholders to help in conflict identification and resolution based on the priorities explicitly expressed by the stakeholders [Brito et al., 2007]. Prior to our work discussed below, there has been no research on text-only based conflict identification in AORE.

This chapter presents EA-Analyzer, a tool for identifying conflicts in textual AO requirements, without needing to convert the textual artifacts into an alternative format, or engaging stakeholders directly. The tool operates on annotated natural language text and compositions defined using the RDL annotations [Chitchyan et al., 2007]. The annotations do not alter or reduce the textual requirements, but only decorate text with syntactic and semantic linguistics tags [Chitchyan, 2007]. A Bayesian learning method, called Naive Bayes [Mitchell, 1997], is utilized by EA-Analyzer to learn the nature of the composed concerns and to detect conflicts within the textual specifications.

This chapter is centered on a case-study based discussion of the accuracy of the tool, where EA-Analyzer is applied to the Crisis Management System, a case study used as an established benchmark in several areas of aspect-oriented research. The initial evaluation of the tool suggests that this is a promising direction for text-based conflict identification.

The rest of this chapter is organized as follows. Section 2 presents the related work and discusses the advantages and disadvantages of aspect-oriented approaches when compared to EA-Analyzer. Section 3 details the approach and the EA-Analyzer tool developed to address the problems discussed in the pervious section. Section 4 presents the case study demonstrating the evaluation of the tool. Section 5 concludes the chapter.

2 Related Work

The most popular approaches that deal with conflicts in requirements are the goal-oriented and aspect-oriented ones; hence, Section 2.1 presents some related work on goal-oriented approaches and Section 2.2 discusses the related approaches of aspect-oriented requirements engineering, where EA-Analyzer is a novel approach within this research area.

2.1 Goal-Oriented Approaches

In the NFR framework [Chung et al., 1999], the focus is on the identification of conflicts of non-functional requirements — it does not explicitly deal with

functional concerns, but establishes a link to them. The analysis starts with softgoals, i.e. quality attributes of a system. The system's softgoals may be security, usability, performance and availability. In the NFR framework, softgoals are normally decomposed and refined into more solution space model elements, captured by a softgoal graph structure. By analyzing the graph, interfering softgoals can be found, e.g. security goals interfere with usability in general. Resolution of such conflicts is achieved by selecting the most appropriate softgoals after some trade-off analysis.

*i** [Yu, 1995] was developed for modeling and reasoning about organizational environments and their information systems. It focuses on the concept of intentional actor. *i** has two main modeling components: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. The SD model describes the dependency relationships among the actors in an organizational context. The SR model provides a more detailed level of modeling than the SD model, since it focuses on the modeling of intentional elements (goals, softgoals, tasks and resources) and relationships internal to actors. Intentional elements are related by means-end or decomposition links. Means-end links are used to specify alternative ways to achieve goals. Decomposition links are used to decompose tasks. Apart from these two links, there are the contribution links, which can be positive or negative. These are the basis for the conflict identification, which is specified in a similar way to the NFR framework. In both approaches, the conflict degree is specified and alternatives are used to solve conflicts.

KAOS [van Lamsweerde et al., 1991] is a systematic approach for discovering and structuring system level requirements. In KAOS, goals can be divided into requirements (a type of goal to be achieved by a software agent), expectations (a type of goal to be achieved by an environment agent) and softgoals (e.g., quality attributes). In KAOS, goals can be refined into subgoals through and/or decompositions. There is also the possibility of identifying conflicts between non-functional goals and represent it in the goal models.

2.2 Aspect-Oriented Requirements Engineering Approaches

Aspect-Oriented Requirement Engineering (AORE) approaches have enabled the early identification of candidate crosscutting concerns within problem domains. Such strategies enable requirements engineers to specify how requirements compose with one another to explicitly externalize their interdependencies.

This has significant advantages for reasoning about requirements, as their mutual influences and tradeoffs can be identified before architecture is derived. As well as this, the transition to an aspect-oriented architecture can be eased by the explicit recognition of early aspects within the domain.

However, this benefit also brings with it a significant challenge - namely, the accurate detection of conflicts between requirements. The increased modularity

and advanced composition mechanisms which AORE approaches tend to employ can complicate the task of discerning where requirements interact with one another and, whether a given interaction constitutes a potential conflict. This issue has received a great deal of research attention within the AO community when the conflict is expressed at the code level; but research at the requirements level is much less mature. In this section we discuss the existing AORE approaches that support conflict detection, and highlight the open issues in this area. We group the available AORE approaches on basis of their overall conflict identification strategy into the following three groups:

Formalization-Based approaches

Within the AO conflict detection research area, many current approaches require some formal specification of requirements in order to detect conflicts among requirements. In other words, these approaches require precise expression of the properties of requirements and decide whether the compositions specified over these requirements invalidate these properties.

Examples of this strand of work are the AO Composition Frames [Laney et al., 2004]; Composition Frames model the semantics of requirements (in the form of Problem Frames) being composed with one another. The requirements of this composition - that is, the formal properties of its satisfaction - can be validated against the state machine expressed in the Composition Frame, and thus conflicts detected. Here the validity of the conflict detection depends on the sound construction of the Problem Frames and their compositions.

In [Mostefaoui and Vachon, 2007], AO models are specified in Aspect-UML, which includes formal annotations of aspects and joinpoints. These Aspect-UML models are transformed into Alloy, a structural modeling language based on first-order logic. Alloy includes an analyzer that can check the validity of assertions over a model, and so the Aspect-UML model of an AO system can be checked for aspects introducing properties to the system that render other aspect assumptions invalid, and thus determine conflicts.

Similarly, the work in [Weston et al., 2008] presents a conflict detection technique based on transformation of textual compositions into temporal logic formulae based on a catalogue of formalizations of natural language operators. The semantics of the compositions can thus be compared with one another for temporal overlap and violation of system properties, which implies a conflict between requirements.

The major disadvantage of these approaches is that the transformation of requirements into specific formal representations will require substantial time and effort, which may outweigh the advantages of precisely detecting conflicts. Moreover, the formalized representations become less accessible to broader audiences. For instance, in order to understand implications of the Alloy analyzer results, the analyst has to be familiar with the formalization framework. Moreover,

if there are any errors introduced in the formalization process, the detected conflicts may not be truly representative of those present in the requirements themselves.

Model-based approaches

A number of AORE approaches take a (design-level) model-based view on conflict detection; that is, they expect the requirements to be (at least initially) structured into specific models before conflicts can be detected.

For instance, the work in [Mehner et al., 2006] models requirements as use cases in UML notation, and the crosscutting concerns are activities which refine the use cases. The approach then translates these UML diagrams into type graphs, with activities being modeled as graph transformations. Applying these graph transformations sequentially can thus reveal conflicts between requirements. A similar technique based on statechart weaving on UML models was proposed in [Shaker and Peters, 2006].

Similarly, the work in [Barais et al., 2008] adapts the Theme/UML [Baniassad and Clarke, 2004] approach to formally model compositions between base and aspect concerns. Certain forms of conflict based on global properties, such as visibility and kind, can then be discerned and automatically resolved. Another similar technique for class diagrams is presented in [Reddy et al., 2006].

The disadvantages of the model-based approaches are twofold. Firstly, the necessity of modeling adds an extra step to the conflict detection process, which may require additional time and effort. Secondly, the structuring of requirements into models may lose information, which means that information encoded in the requirements, including potential conflicts, may be omitted/lost before the interaction analysis commences. Also, similar to formalization-based approaches, a modeling error may invalidate the results of the analysis.

Stakeholder priority-based approaches

Finally, the stakeholder priority-based work [Moreira et al., 2005; Brito and Moreira, 2003; Brito and Moreira, 2004; Rashid et al., 2003] handles conflicts via stakeholder involvement. If interactions can be identified using a technique such as ARCADE [Rashid et al., 2003], the stakeholders can then determine whether such compositions are positive, negative or neutral from their point of view, and refine the requirements accordingly [Rashid et al., 2003]. Alternatively, stakeholders state their preferred non-functional requirements up-front, and mathematical reasoning techniques (i.e., a multi-criteria decision making method called Analytical Hierarchy Process [Saaty, 1980; Saaty, 2008]) are then applied to help conflict resolution [Brito et al., 2007].

More recently, in the AMPLE project [AMPLE, 2011], a novel hybrid assessment method, HAM, was proposed and a software tool was developed. HAM combines the best properties of two well known multi-criteria decision making methods, the Analytical Hierarchy Process and the Weighted Average [Triantaphyllou, 2000]; this combination helped to avoid some problematic features of those methods [Ribeiro et al., 2011].

The main limitations of these approaches are that: (i) each concern must be allocated a specific priority; (ii) conflict handling is often based on one criterion, the priority (except for [Brito et al., 2007], where multi-criteria analysis is supported); (iii) the conflict identification and resolution requires direct involvement of the stakeholders.

In summary, although the above discussed AORE approaches can help in conflict identification for AORE, what is missing from the current state of the art is a tool-supported informal approach which is able to determine potential interactions based on compositions of the requirements themselves, without having to resort to the formalization / modeling or the subjective (and frequently arbitrary) opinions of stakeholders. Such a tool would enable conflicts to be detected quickly from textual specifications themselves, and thus provide a cost-effective solution to developers.

3 Detecting Conflicts in an Aspect-Oriented Specification

This section presents the EA-Analyzer tool and the process utilized to identify conflicts between requirements in the Crisis Management System. We will start presenting the annotation process of the Crisis Management specification with the Requirements Description Language (RDL). The following sections describe the inner workings of the tool on the annotated specification and an empirical evaluation of the tool.

3.1 Annotating Textual Requirements with RDL

The Requirements Description Language (RDL) [Chitchyan, 2007] utilizes XML tags to annotate a natural language specification, in order to express dependencies and interactions between various groups of requirements (such as viewpoints and use cases). A previous chapter of this book presents a detailed description of the RDL and discusses the usability of the approach; hence, we refer the reader to this chapter for a detailed discussion regarding the RDL.

Figures 1 and 2 show an example of a Non-Functional Requirement (NFR) in the Crisis Management System that has been annotated with the RDL tags. The annotated RDL text is generated with the semi-automated EA-Miner [Sampaio et al., 2005] tool, which is based on a general purpose NLP tool, Wmatrix [Rayson, 2010].

```
<Concern name="Real-time">
…
<Requirement id="3">
    The <Subject>system</Subject>
    <Degree type="modal" semantics="obligation" level="high">shall</Degree>
    be able to
    <Relationship type="Move" semantics="Transfer_Posession">retrieve</Relationship>
    any stored
    <Object>information</Object>
    with a maximum
    <Object>delay</Object> of 500 milliseconds.
  </Requirement>
 </Concern>
```

Fig. 1 Example of a NFR requirement in the Crisis Management System

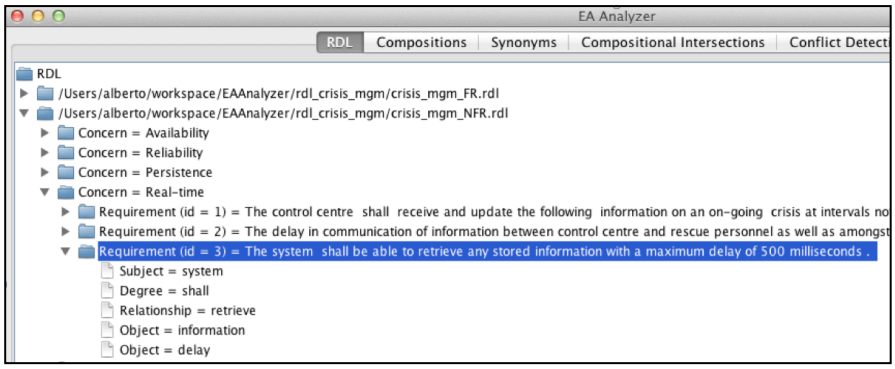

Fig. 2 Visualizing the NFR requirement in EA-Analyzer

In addition, the RDL tags also express dependencies and interactions between requirements. Hence, an analyst can define domain relationships (via RDL compositions) using only the natural language text. For instance, RDL compositions can mandate that a requirement must precede another one, such as the real-time requirement in Figure 1 (“The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds”), which should be satisfied before any other requirement that retrieves information.

An RDL composition consists of three parts, namely *Constraint*, *Base*, and *Outcome*. Each part has a semantic query that selects requirements from the specification with the aim of ensuring a desired interaction. For instance, Figure 3 presents a composition that must ensure that the requirements selected by the *Base* query (e.g., "The system shall have access to detailed maps, terrain data and weather conditions …" in Figure 4) are constrained by the requirements selected by the *Constraint* query (i.e., "The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds").

```
<Composition name="Performance (Real-time)">
        <Constraint operator="apply">(subject="system" and relationship="retrieve" and
        object="information" and object="delay")</Constraint>
        <Base operator="before">relationship="handling" or relationship="processing" or
        relationship="request" and relationship="access"</Base>
        <Outcome operator="ensure"/>
</Composition>
```

Fig. 3 Example of a Composition in the Crisis Management System

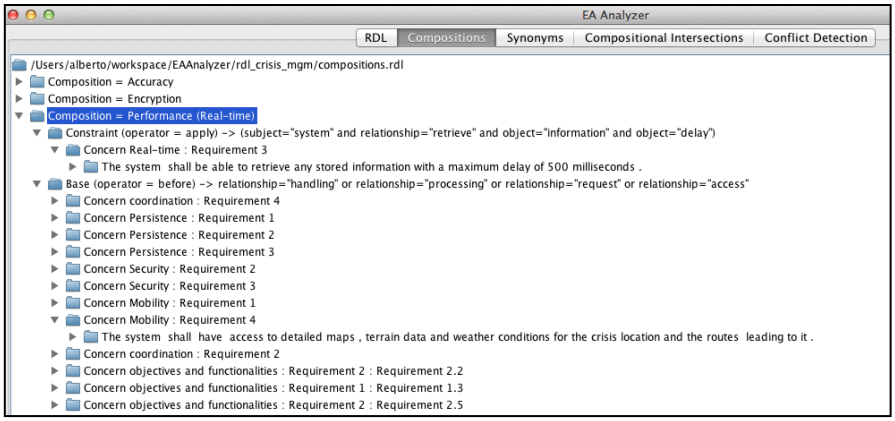

Fig. 4 Visualizing the Composition in EA-Analyzer

3.2 Detecting Conflicts in the Crisis Management Specification

The main goal of EA-Analyzer is to detect conflicts within a textual specification that has been previously annotated with RDL tags; recall that RDL tags are added with the help of the EA-Miner tool. In addition, the tool has a Graphical User Interface (GUI) that helps to visualize the annotated specification and the composition, such as the examples in Figure 2 and 4.

In EA-Analyzer, the problem of detecting conflicts is formulated as a classification problem, which is a well-studied problem in machine learning [Mitchell, 1997]. The tool operates on RDL by using its compositions and annotated requirements, and utilizes composed requirements to decide whether they have a conflicting dependency.

EA-Analyzer has to go through a learning process before the tool can be utilized for detecting conflicts. The learning process consists of the following steps: (i) Identifying all the sets of requirements that crosscut one or more base concerns, also known as Compositional Intersections (Section 3.2.1); (ii) Generating training examples for the learning method by labeling the Compositional Intersections (Section 3.2.2); and, (iii) Training the classifier based on the examples generated in step (ii) (Section 3.2.3).

3.2.1 Identifying Compositional Intersections

The first step in the learning process is concerned with the identification of the compositional intersections; compositional intersections are used as a basis to detect conflicts among composed concerns, because they explicitly represent the interactions of a requirement with other requirements with reference to a base requirement.

A compositional intersection is the union of all the constraint requirements (i.e., requirements that have been selected by the constraint queries) that crosscut the same base requirement. For instance, the Crisis Management specification has a composition that selects the constraint requirement R_3 of the *Real-time concern* (i.e., "The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds") and the base requirement R_1 of the *Persistence concern* (i.e., "The system shall provide support for storing, updating and accessing the following information…"). In addition, the specification has another composition that selects the constraint requirement R_5 of the *Security concern* (i.e., "All communications in the system shall use secure channels compliant with AES-128 standard encryption") and the aforementioned base requirement (i.e., R_1 of the *Persistence concern*). Hence, R_3 of the *Real-time concern* and R_5 of the *Security concern* are part of the compositional intersection of R_1 of the *Persistence concern* as shown in Figure 5.

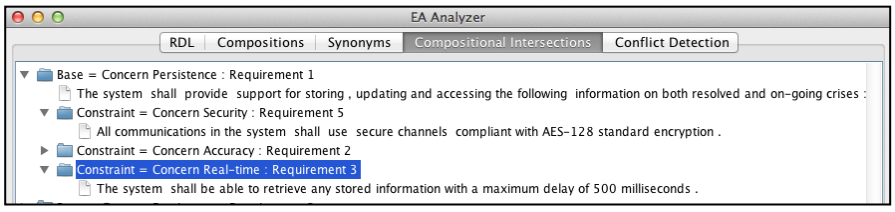

Fig. 5 Visualizing the Compositional Intersections in EA-Analyzer

3.2.2 Generating Training Examples

The machine learning technique utilized by EA-Analyzer requires a set of labeled examples to train the tool. This step enables the tool to be trained on a per-organization basis, so that each organization can have their EA-Analyzer tool tailored for detecting conflicts in their requirements documents.

Labeled examples are time-consuming to obtain since they normally require a human annotator to examine and label each training example. In order to reduce this burden, we have implemented a module in EA-Analyzer that partially automates this step. Figure 6 presents the user interface (UI) that helps the human annotator label the composition intersection that has been previously identified (Section 3.2.1). In the UI, the human annotator is only required to select the conflicting requirements from the top list and the tool automatically performs a brute force procedure that labels each occurrence of the conflicting dependency in the set of examples (i.e., the compositional intersection that have been selected to train the tool). Figure 6 shows also that the tool requires not only training examples of conflicting dependencies within a compositional intersection but also examples of requirements within a compositional intersection that are interacting harmoniously.

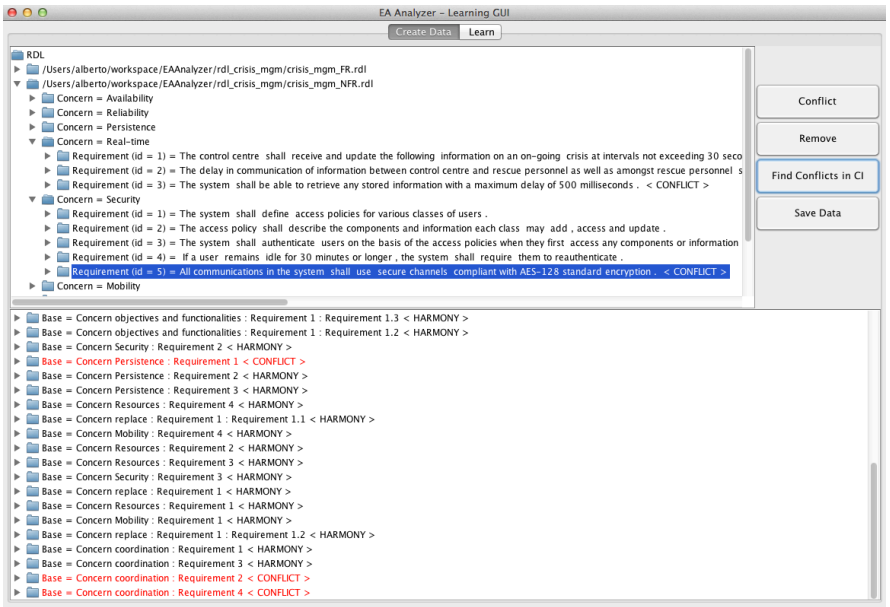

Fig. 6 Generating Training Examples with EA-Analyzer

Figure 6 presents a well-known example of a potential conflict between an *Encryption* requirement and a *Performance* requirement [Sampaio et al., 2007], since introducing encryption into a system reduces its responsiveness. The *Encryption* requirement is R_5 of the *Security concern* (i.e., "All communications in the system shall use secure channels compliant with AES-128 standard encryption") and the *Performance* requirement is R_3 of the *Real-time concern* (i.e., "The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds"). In this example, the human annotator has to select these two conflicting requirements from the top list and the tool automatically labels each occurrence of the conflicting dependency in the compositional intersections below. The labeled examples are then saved to a file so that the tool can train the machine learning technique.

3.2.3 Training EA-Analyzer to Identify Conflicts

EA-Analyzer utilizes the Naïve Bayes learning method to train the tool based on the training examples provided in the previous step (Section 3.2.2). The learning method leads to a bag of words model (BoW); the BoW is a method in Natural Language Processing that models text as an ordered collection of independent words represented in a term-frequency vector, disregarding grammar[5] and even word order.

For instance, one can imagine the BoW of EA-Analyzer with two bags full of words. The first bag is filled with words found in compositional intersections that have a potential conflict, such as the potential conflict presented in Section 3.2.2 (i.e., the well-known example of a potential conflict between an *Encryption* requirement and a *Performance* requirement). The second bag is filled with words found in compositions that do not have a potential conflict. While some words can appear in both bags, the first bag will contain conflict-related words such as "encryption" and "retrieve" much more frequently. On the other hand, the second bag will contain more words related to the other requirements. Hence, a new compositional intersection that has more words that come from the first bag than the second bag will be classified as a conflict.

[5] Please note that grammar and semantics are used in RDL composition definitions, as discussed previously. Thus, they are indispensable in the task of collecting the required bags or words. Once such words are collected, in the EA-Analyzer learning phase, the grammar and semantics are not used any further.

3.2.4 Advantages and Disadvantages of the learning method of EA-Analyzer

This learning method in EA-Analyzer presents two advantages. First, the learning method only requires a small amount of data to train the Naive Bayes classifier [Mitchell, 1997]. Second, the learning method can be easily trained on a per-organization basis, so that each organization can have their EA-Analyzer tool tailored for detecting conflicts in their requirements documents. Moreover, it has been proven to be very powerful (and with outstanding performance) in NLP problems such as text classification and topic modeling. However, the main disadvantage of this learning method is that it only considers the distribution of the words and loses the relationships between them. To overcome this problem, search engines commonly use vocabularies consisting of combinations of words or expressions, and the same technique is used in EA-Analyzer.

In EA-Analyzer, the binary classification of a compositional intersection as either harmony or conflict could be perceived as an over-simplification of requirements' relationships. The relationship of two quality requirements could be considered conflicting in one system and tolerable in another by a human analyst. However, EA-Analyzer will always pinpoint the potential presence of such conflicts. It is then up to the requirements analyst to consider if a given potential conflict can be tolerable in a given context, and so disregard it from the set of real conflicts for that system. Such classifications are not directly supported by the conflict identification support of EA-Analyzer; we consider these to constitute the follow-up step of conflict resolution.

4 Empirical Evaluation

This section presents an empirical evaluation of the tool, where the main goal was to assess the ability of EA-Analyzer to detect conflicts using training data gathered from four different documents, each representing a different domain. The documents were selected based on their suitability for this evaluation, with selection criteria including: domain, requirement type, complexity and use in previous studies. In addition, three documents originate from industrial organizations and the fourth document is a case-study extensively used in academia to evaluate AO modeling techniques. Furthermore, each of these documents was created prior to the conception of this study by external personnel. The four documents selected were:

- *Health Watcher* (HW) [Soares et al., 2006] is a web based health support system which the public can use to register health-related complaints and query disease and symptom information.

- *Smart Home* (SH) [Pohl et al., 2005] is an embedded system which provides functionality to control various sensors and actuators around the home (i.e., lights, blinds, heating, etc.).
- *CAS* [Ayed and Genssler, 2009] is a customer relationship management application (CRM) which utilizes service mash-ups and mobility support in a hosted software-as-a-service environment.
- *Crisis Management System* (CM) [Kienzle et al., 2010] is a crisis management system for emergency situations (e.g., natural disasters, accidents, terrorist attacks).

The evaluation consists of four experiments, in which we utilized each requirements document (HW, SH, CAS and CM) in turn as the training set and evaluated the classification accuracy of the tool with the other three documents. Table 1 shows some characteristics of the four documents selected for this study, and the characteristics present two different dimensions of the requirements specifications: (i) the size of the documents, by showing the number of words, compositions and compositional intersections (CI); and (ii) the number of compositional intersections that have the *Encryption—Performance* conflict. Each experiment used the *Encryption—Performance* conflict to evaluate the classification accuracy of the tool, because it is the only NFR conflict type that occurs in all four documents.

	HW	SH	CAS	CM
Words in RDL	1764	4699	1053	5961
Num. of Compositions	17	9	5	8
Num. of CI	89	71	16	43
Num. of CI with *Encryption – Performance* conflict	23	5	3	16

Tab. 1 Results of the classification accuracy in each experiment

Table 2 presents the classification accuracy of the tool with the four different training sets. The classification accuracy of the HW and CM documents is 93.90%, while the experiment with SH document achieved 92.05% and the CAS experiment yielded a classification accuracy of 48.51%. All the results are compared to a baseline accuracy of 50%, as randomly assigned classes should yield an approximate 50% accuracy. The results that use HW, SH and CM as training sets yield classifications results above the baseline accuracy; however, the experiment with the CAS document yields a classification result below the baseline accuracy. This may suggest that the size of the training set (the CAS document has only 16 compositional intersections – see Table 1) can significantly influence the classification accuracy of the tool. Despite the poor result with the CAS document, the results with the other three document, when a larger number

of examples are utilized to train the tool, present very high classification results. This suggests that the machine learning technique in EA-Analyzer is capable of detecting conflicts in aspect-oriented specifications. A more extensive and detailed evaluation of the tool can be found in [Sardinha et al., 2013].

	Training Sets			
Validation Data	HW	SH	CAS	CM
HW		88.64%	34.09%	100.00%
SH	94.20%		100.00%	94.20%
CAS	87.50%	87.50%		87.50%
CM	100.00%	100.00%	11.43%	
Weighted Average	93.90%	92.05%	48.51%	93.90%

Tab. 2 Results of the classification accuracy in each experiment

5 Conclusions

The AO approach is an effective way to modularize and compose concerns in requirements specifications. In addition, AORE methods help to externalize interactions and interdependencies between concerns by utilizing explicitly dedicated composition specifications. These composed concerns are an excellent starting point for detecting conflicts within the requirements specification. However, detecting conflicts in large natural language specifications can be a burden for a requirements engineers, due to the large number and complexity of the interdependencies to be considered. As discussed earlier, the approaches based on formal specifications, models and stakeholder priorities, developed to date in the AORE community, are unable to provide low effort and high precision techniques for conflict identification in large AO specifications.

This chapter presents the EA-Analyzer tool, in which we demonstrate that it is indeed possible to automate the process of detecting conflicts within textual AO requirements specifications. In addition, we present an empirical evaluation of the tool with three industrial-strength requirements documents and a well established academic case-study used in the AO research community. The results show that conflicts within requirements specifications can be detected with a high accuracy, as longs as a sufficient number of examples is utilized in the training set.

As future work, we will focus efforts on the empirical evaluation of the tool with other requirements documents from different domains to validate the generalization power of the learning method in EA-Analyzer. In addition, we will also test a number of other classifiers in the tool, such as SVM [Bishop, 2006] and nearest-neighbor methods [Bishop, 2006]. The utilization of different machine learning classifiers may helps us identify the best machine learning approach for detecting conflicts.

EA-Analyzer is the first tool for automated conflict identification in textual AO requirements and compositions, and this work demonstrates that the power of AORE to represent concern interrelationships knowledge can be effectively harvested for conflict detection within natural language specifications. Hence, we see this work as the stepping stone towards effort reduction in AORE conflict identification, and supporting application of advanced modularity and analysis in textual requirements.

References

1. AMPLE project: http://www.ample-project.net (2011)
2. Ayed, D., Genssler, T.: Dynamic variability in complex, adaptive systems. Deliverable D6.1 of DiVA EC project (2009)
3. Baniassad, E., Clarke, S.: Theme: An approach for aspect-oriented analysis and design. In: ICSE'04: Proceedings of the 26th International Conference on Software Engineering. IEEE Computer Society, Washington (2004)
4. Barais, O., Klein, J., Baudry, B., Jackson, A., Clarke, S.: Composing multi-view aspect models. In: ICCBSS'08: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008). IEEE Computer Society, Washington (2008)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
6. Brito, I., Moreira, A.: Towards a Composition Process for Aspect-Oriented Requirements. presented at Early Aspects Workshop at AOSD'03, Boston, USA (2003)
7. Brito, I., Moreira, A.: Integrating the NFR Approach in a RE Model, presented at Early Aspects Workshop at AOSD'04, Lancaster, UK (2004)
8. Brito, I.S., Vieira, F., Moreira, A., Ribeiro, R.: Handling conflicts in aspectual requirements compositions. Transactions on Aspect Oriented Software Development (TAOSD) (2007)
9. Chitchyan, R.: Semantics-based composition for aspect-oriented requirements engineering. Ph.D. thesis, Computing Department, Lancaster University (2007)
10. Chitchyan, R., Rashid, A., Rayson, P., Waters, R.: Semantics-based composition for aspect-oriented requirements engineering. In: AOSD'07: Proceedings of the 6th International Conference on Aspect-Oriented Software Development. ACM, New York (2007)
11. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional Requirements in Software Engineering. Kluwer Academic, Dordrecht (1999)
12. Kienzle, J., Guelfi, N., Mustafiz, S.: Crisis management systems: A case study for aspect-oriented modeling. In: Katz, S., Mezini, M., Kienzle, J. (eds.) Transactions on Aspect-Oriented Software Development VII. Lecture Notes in Computer Science, vol. 6210, pp. 1–22. Springer, Berlin (2010)
13. Laney, R., Barroca, L., Jackson, M., Nuseibeh, B.: Composing requirements using problem frames. In: RE'04: Proceedings of the Requirements Engineering Conference, 12th IEEE International. IEEE Computer Society, Washington (2004)
14. Mehner, K., Monga, M., Taentzer, G.: Interaction analysis in aspect-oriented models. In: RE'06: Proceedings of the 14th IEEE International Requirements Engineering Conference. IEEE Computer Society, Washington (2006)
15. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
16. Moreira, A., Araújo, J., Rashid, A.: Multi-Dimensional Separation of Concerns in Requirements Engineering. In: International Conference on Requirements Engineering (RE), Paris, France, (2005)

17. Mostefaoui, F., Vachon, J.: Design-level detection of interactions in aspect-UML models using Alloy. J. Object Technol. 6(7), 137–165 (2007)
18. Pohl, K., Bockle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, New York (2005)
19. Rashid, A., Moreira, A., Araújo, J.: Modularisation and composition of aspectual requirements. In: AOSD'03: Proceedings of the 2nd International Conference on Aspect-Oriented Software Development. ACM, New York (2003)
20. Rayson, P.: Wmatrix. http://www.comp.lancs.ac.uk/ucrel/wmatrix/ (2010)
21. Reddy, Y.R., Ghosh, S., France, R.B., Straw, G., Bieman, J.M., McEachen, N., Song, E., Georg, G.: Directives for composing aspect-oriented design class models. In: Trans. Aspect-Oriented Software Development, pp. 75–105 (2006)
22. Ribeiro, R., Moreira, A., Broek, P., Pimentel, A.: Hybrid Assessment Method for Software Engineering Decisions. Decision Support Systems 51(1), 208-219 (2011)
23. Sampaio, A., Chitchyan, R., Rashid, A., Rayson, P.: EA-Miner: A tool for automating aspect-oriented requirements identification. In: ASE'05: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. ACM, New York (2005)
24. Sampaio, A., Greenwood, P., Garcia, A.F., Rashid, A.: A comparative study of aspect-oriented requirements engineering approaches. In: ESEM'07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society, Washington (2007)
25. Sardinha, A., Chitchyan, R., Weston, N., Greenwood, P., Rashid, A.: EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements. Automated Software Engineering 20(1), 111-135 (2013)
26. Saaty, T.: Decision making with the analytic hierarchy process. Int. J. Services Sciences 1, 83-98 (2008)
27. Saaty, T.: The Analytic Hierarchy Process. McGraw-Hill (1980)
28. Shaker, P., Peters, D.K.: Design-level detection of interactions in aspect-oriented systems. In: Proceedings of the Aspects, Dependencies, and Interactions Workshop at ECOOP 2006 (2006)
29. Soares, S., Borba, P., Laureano, E.: Distribution and persistence as aspects. Softw. Pract. Exp. 36(7), 711– 759 (2006)
30. Triantaphyllou, E.: Multi-Criteria Decision Making Methods: A Comparative Study. Kluwer Academic Publishers (2000)
31. van Lamsweerde, A., Dardenne, A., Delcourt, B., Dubisy, F.: The KAOS project: Knowledge acquisition in automated specification of software. In: Proceedings AAAI Spring Symposium Series, Stanford University. American Association for Artificial Intelligence, Washington (1991)
32. Weston, N., Chitchyan, R., Rashid, A.: A formal approach to semantic composition of aspect-oriented requirements. In: RE'08: Proceedings of the 16th International Requirements Engineering Conference (2008)
33. Yu, E.: Modelling Strategic Relationships for Process Reengineering. Ph.D. Thesis. Dept. of Computer Science, University of Toronto (1995)