2009 IEEE/ACM International Conference on Automated Software Engineering

EA-Analyzer: Automating Conflict Detection in Aspect-Oriented Requirements

Alberto Sardinha, Ruzanna Chitchyan, Nathan Weston, Phil Greenwood, Awais Rashid
Computing Department
Lancaster University
Lancaster, LA1 4WA, UK
{*sardinha, rouza, westonn, greenwop, awais*}*@comp.lancs.ac.uk*

***Abstract*—One of the aims of Aspect-Oriented Requirements Engineering is to address the composability and subsequent analysis of crosscutting and non-crosscutting concerns during requirements engineering. Composing concerns may help to reveal conflicting dependencies that need to be identified and resolved. However, detecting conflicts in a large set of textual aspect-oriented requirements is an error-prone and time-consuming task. This paper presents EA-Analyzer, the first automated tool for identifying conflicts in aspect-oriented requirements specified in natural-language text. The tool is based on a novel application of a Bayesian learning method that has been effective at classifying text. We present an empirical evaluation of the tool with three industrial-strength requirements documents from different real-life domains. We show that the tool achieves up to 92.97% accuracy when one of the case study documents is used as a training set and the other two as a validation set.**

***Keywords*-Aspect-Oriented Requirements Engineering; Aspect-Oriented Software Development; Conflicting Dependencies; Requirements Analysis; Requirements Composition.**

I. Introduction

Aspect-Oriented Requirements Engineering (AORE) [9] aims to address the composability and subsequent analysis of crosscutting and non-crosscutting concerns. In AORE, a concern encapsulates one or more requirements related to a certain matter of interest. For example, a security concern may contain a data encryption requirement and a security check requirement. Concerns that crosscut other concerns are called crosscutting concerns or aspects (e.g., security, distribution, and performance).

Compositions are used to explicitly represent and analyze the interdependencies between concerns. The composed concerns are also utilized as a basis for detecting potential conflicts between concerns before architecture is derived. In some cases, the detection of a conflicting dependency also reveals the tradeoff of a development technique in a very early stage of the software life cycle [3].

Many AORE approaches [4] [7] [12] for detecting conflicts require some formal specification of requirements. The main disadvantage of these approaches is that the transformation of textual requirements into specific formal representations may require substantial time and effort. Our experience suggests that the analysis of conflicting dependencies in textual requirements is still being done manually with visual inspection. Furthermore, the detection of conflicts in large textual specifications is an error-prone and time-consuming task that creates a burden on the requirements engineer.

This paper presents EA-Analyzer, the first automated tool for identifying conflicts in textual aspect-oriented requirements. The tool operates on RDL specifications; RDL [2] provides an annotation mechanism for large textual specifications which enables the definition of compositions based on semantic natural-language operators (see example in Figure 2). A Bayesian learning method, called Naive Bayes [5], is utilized by the tool to learn the nature of the composed concerns and to detect conflicts within a RDL specification. We evaluated the tool with three industrial-strength requirements documents from different real-life domains: a Web-based application that manages health-related complaints [11], a home automation system [8], and a customer relationship management application [1]. The results show that it achieves up to 92.97% accuracy when using one of the documents as a training set and the other two as a validation set.

This paper is organized as follows. Section II describes the learning method for detecting conflicts and the EA-Analyzer tool, one of the key contributions of this paper. Section III presents the empirical evaluation of the tool, another key contribution of this paper. Finally, the conclusions are presented in Section IV.

II. EA-Analyzer

EA-Analyzer is a tool for detecting conflicts within a textual aspect-oriented requirements document. In our approach, the problem of detecting conflicts is formulated as a classification problem, which is a well-studied problem in machine learning [5]. The tool operates on RDL by using its compositions and natural-language requirements, and utilizes composed concerns to decide whether they have a conflicting dependency among requirements.

In order to detect conflicts with EA-Analyzer, the following steps are required: (i) Identify all the sets of concerns that crosscut one or more base concerns, also known as Compositional Intersections (Section II-A); (ii) Generate training examples for the learning method by labeling the Compositional Intersections (Section II-B); and, (iii) Train

1527-1366/09 $29.00 © 2009 IEEE
DOI 10.1109/ASE.2009.31

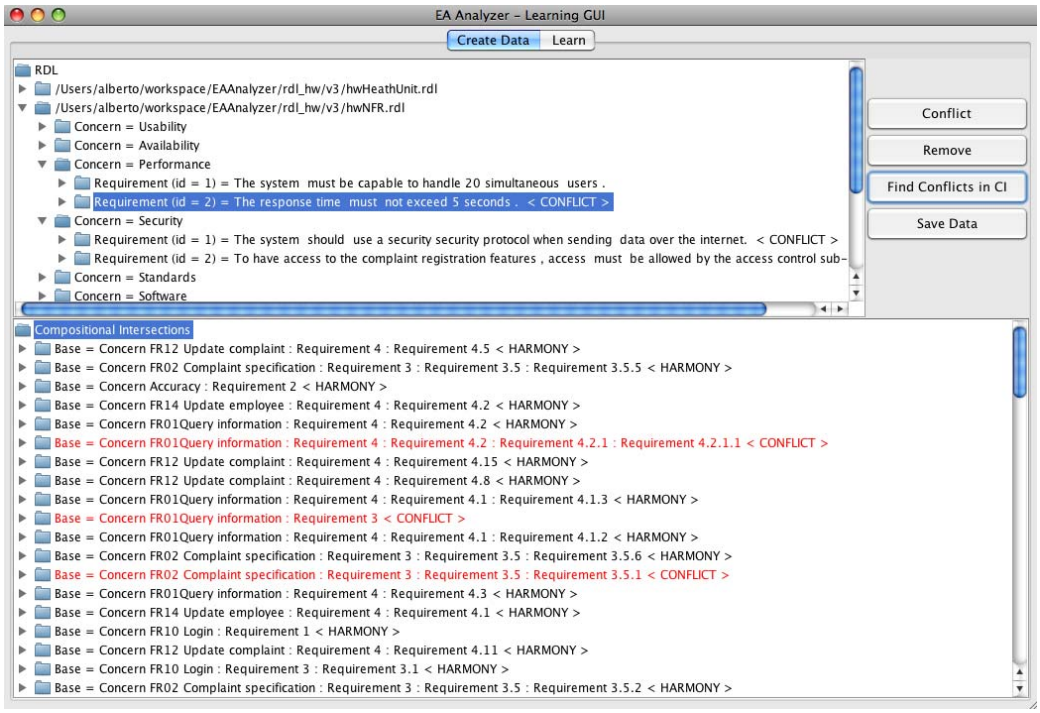

Figure 1. The User Interface for Generating Training Examples

the classifier based on the examples generated in step (ii) (Section II-C).

A. Identifying Compositional Intersections

Compositional Intersections are used as a basis to detect conflicts among composed concerns, because they explicitly represent the interactions of a concern with other concerns with reference to a base requirement. This Section describes the algorithm used to identify Compositional Intersections, which is a modified version of the algorithm in [6].

Let $C_1, C_2, C_3, ..., C_n$ be concerns in the system requirements and $R_{i,j}$ be the requirement i encapsulated by concern C_j. The compositions in the aspect-oriented specification describe how a set of constraint requirements, $Cr_k = \{R_{i,j}|Q_k^{Cr}\}$, crosscut a set of base requirements, $Br_k = \{R_{i,j}|Q_k^{Br}\}$, where Q_k^{Cr} and Q_k^{Br} are, respectively, the constraint and base query of composition k.

Let $Sc_{i,j}$ be the set of compositions where $R_{i,j}$ is a base requirement. Thus, the Compositional Intersection of requirement $R_{i,j}$ is defined by equation 1.

$$CI_{i,j} = \bigcup_{k \in Sc_{i,j}} Cr_k \quad (1)$$

A Compositional Intersection is the union of all the constraint requirements that crosscut the same base requirement. Figure 2(a) presents an example of a composition in RDL from the Health Watcher system [11] that selects the constraint requirement $R_{1,1}$ = "The system should use a security protocol when sending data over the internet" to crosscut the following base requirements: $R_{2,1}$ = "The login and password are sent to the server", $R_{2,2}$ = "The conclusion is sent to the server", and $R_{2,3}$ = "The entered data is transmitted to the server". A second composition from the Health Watcher system selects another constraint requirement $R_{3,1}$ = "The response time must not exceed 5 seconds" to crosscut the base requirement $R_{2,3}$ = "The entered data is transmitted to the server". Thus, the compositional intersection of $R_{2,3}$ is $\{R_{1,1}, R_{3,1}\}$. So for each base requirement in the specification, we can find a Compositional Intersection.

```
<Composition name="Secure Protocol Composition">
      <Constraint operator="apply">relationship="use" and
      object="security protocol"</Constraint>
      <Base operator="concurrent">relationship="send" or
      relationship="receive"</Base>
      <Outcome operator="satisfy">relationship="raise" and
      object="error message"</Outcome>
</Composition>
```

(a) Composition enforcing use of a security protocol (i.e., encryption protocol) when sending/receiving data over internet

Concern Login: Requirement "The login and password are *sent* to the server."
Concern Update Complaint: Requirement "The conclusion is *sent* to the server."
Concern Register New Employee: Requirement "The entered data is *transmitted* to the server."
…

(b) A set of requirements selected by the base query from 2(a)

Figure 2. An Example of the RDL from the Health Watcher System

B. Generating Training Examples

In many classifiers, such as the Bayesian learning method in EA-Analyzer, labeled examples are used to estimate a target function that maps an input vector of features into classes. The features in our classification problem are extracted from the requirements in the Compositional

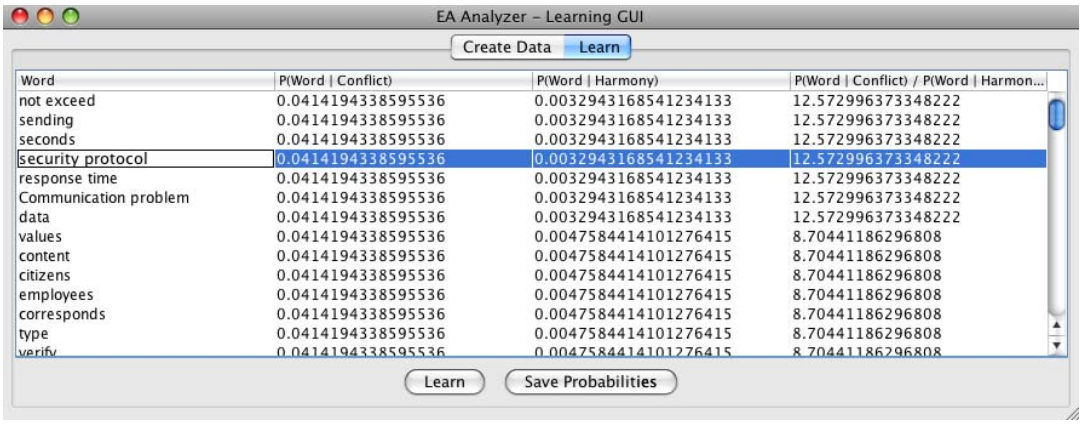

Figure 3. The User Interface for Analyzing the Estimated Probabilities

Intersections. In addition, we have two classes, namely the class of *Conflict*, when two or more requirements present a conflicting dependency, or the class of *Harmony*, when all the requirements are interacting harmoniously.

However, labeled examples are time-consuming to obtain, because they normally require a human annotator to examine and label each training example. Therefore, to reduce the burden on the human annotator, we implemented an interface in the tool that helps a user label each Compositional Intersection and save this training data for the learning process. Figure 1 shows the user interface (UI) in EA-Analyzer that helps a human annotator in the task of labeling the Compositional Intersections. In the UI, the human annotator is only required to select the conflicting requirements from the top list, and the tool automatically labels each Compositional Intersection based on the occurrences of the conflicting dependency.

The example in Section II-A presents a Compositional Intersection of $R_{2,3}$ ("The entered data is transmitted to the server") where $R_{1,1}$ ("The system should use a security protocol when sending data over the internet") and $R_{3,1}$ ("The response time must not exceed 5 seconds") are crosscutting $R_{2,3}$. This is a well-known example of a potential conflict between *Security Protocol*, such as *Encryption*, and *Performance* [10]. In EA-Analyzer, the user is only required to select these requirements, and the tool is made responsible for labeling the Compositional Intersections (i.e., either *Conflict*, if both requirements crosscut the same base, or *Harmony*, otherwise).

C. Training EA-Analyzer to Identify Conflicts

In EA-Analyzer, the problem of detecting conflicts within a Compositional Intersection is formulated as a text classification problem, because the Compositional Intersections are essentially composed of natural-language requirements. The tool is an application of the Naive Bayes [5] classifier, which is an effective approach to the problem of learning to classify text [5].

In this classification problem, the input vector of features is extracted from the text of the requirements, and an estimated target function maps these features into a set of classes $V = \{Conflict, Harmony\}$. Equation 2 shows the Naive Bayes approach to classifying a new instance.

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j) \tag{2}$$

where $< a_1, a_2, ..., a_n >$ is the input vector of features and V is a finite set of classes. The aim of this approach is to assign the most probable target value, v_{NB}, given the input vector of features $< a_1, a_2, ..., a_n >$.

For example, the requirement $R_{1,1}$ = "The system should use a security protocol when sending data over the internet" has the words "security protocol" and "data", and it is also a member of a Compositional Intersection. In order to calculate the most probable class ($Conflict$ or $Harmony$) for this Compositional Intersection, we instantiate Equation 2 as follows:

$$\begin{aligned} v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} & P(v_j) P(a_1|v_j) \\ ...P(a_i = \text{“}security\ protocol\text{”}|v_j) & \\ P(a_j = \text{“}data\text{”}|v_j) & \\ ...P(a_n|v_j) & \end{aligned}$$

The Naive Bayes classifier has a learning step in which the various $P(v_j)$ and $P(a_i|v_j)$ terms are estimated. In the text classification problem, these probabilities are estimated based on the word frequencies over the training data. Equations 3 and 4 are used in EA-Analyzer to estimate the probabilities of Equation 2.

$$P(v_j) = \frac{|CI_j|}{|Examples|} \tag{3}$$

$$P(a_i|v_j) = \frac{n_k + \theta}{n + \theta.|Vocabulary|} \tag{4}$$

where $Examples$ is a set of Compositional Intersections, CI_j is the subset of $Examples$ that are labeled as v_j, $Vocabulary$ is a set of all distinct words w_k that are selected from $Examples$, n is the number of word positions in CI_j, n_k is the number of times a word w_k occurs in class CI_j, and θ is the Laplacian smoothing parameter.

In order to assess the quality of the estimation process, EA-Analyzer has an interface that displays all the words in $Vocabulary$ and the associated probabilities $P(a_i|v_j)$. Figure 3 shows a table with the words sorted by the fraction $\frac{P(a_i|Conflict)}{P(a_i|Harmony)}$, which is a useful method for analyzing the most probable words in the *conflict* class. For example, the estimated probabilities for "security protocol" in Figure 3 are approximately $P(a_i|Conflict) = 0.0414$ and $P(a_i|Harmony) = 0.0033$. Thus, the fraction $\frac{P(a_i|Conflict)}{P(a_i|Harmony)}$ is approximately 12.5730, which means that "security protocol" is 12.5730 more likely to occur in a Compositional Intersection that has been labeled as $Conflict$.

III. Empirical Evaluation

A. Study Configuration

Three documents were selected to be used in this study, all originating from industrial organizations and each representative of different domains. Furthermore, each of these documents were created prior to the conception of this study by industrial personnel. The three documents include:

- *HealthWatcher* [11] is a web-based health support system which the public can use to register health-related complaints and query disease and symptom information.
- *SmartHome* [8] is an embedded system which provides functionality to control various sensors and actuators around the home (i.e., lights, blinds, heating, etc.).
- *CAS* [1] is a customer relationship management application (CRM) which utilizes service mash-ups and mobility support in a hosted software-as-a-service environment.

Each of these applications contain a range of different requirements both functional and non-functional with some types of requirements (e.g. security) occurring across the applications. From previous analysis of these applications they were known to contain requirements that are potentially conflicting. However, no previous work had been undertaken to determine whether these requirements do actually conflict in each of these documents.

Table I shows some characteristics of the three documents selected for this study (HealthWatcher (HW), SmartHome (SH), and CAS). The characteristics present two different aspects of the documents: (i) the size of the RDL, by showing the number of words, compositions and Compositional Intersections (CI); and (ii) the conflict, by showing the number of Compositional Intersections that have the *Encryption - Performance (Enc-Perf)* conflict [10] (*Enc-Perf* is a well-known conflict between non-functional requirements (NFR), because introducing encryption into a system reduces its responsiveness).

	HW	SH	CAS
Words in RDL	1764	4699	1053
Num. of Compositions	17	9	5
Num. of CI	89	71	16
Num. of Enc-Perf	23	5	3

Table I
Characteristics of the Requirements Documents

The hypotheses of this evaluation are the following: firstly, we expect the tool to achieve a high classification accuracy when testing it with different requirements documents from different domains. Secondly, the performance of the tool will differ depending on the training data. Therefore, to test the hypotheses above, we assessed the ability of EA-Analyzer to detect conflicts using training data gathered from different domains.

B. Using a Requirements Document to Learn to Detect Conflicts in Other Documents

In this experiment, we used each requirements document in turn (HW, SH, and CAS) as a training set, and evaluated the classification accuracy of the tool with the other two documents. This is a challenging test for EA-Analyzer, because it shows the tool's ability to generalize from different documents in distinct domains. The vocabulary of each document poses the most problems. As seen in Section II-C, the words in the training set (i.e., the words in the document) are used as features to the classifier. However, some of these words occur in only one of the documents, and this can significantly influence the classification accuracy of the tool. Therefore, to address this issue, we also provide a synonym list to the tool, so that it can match words that have the same meaning across multiple documents.

Tables II - IV present the classification accuracy of the tool with the three different training sets. All the results are compared to a baseline accuracy of 50%. This is due to the fact that randomly assigned classes should yield an approximate 50% accuracy. In Table II, EA-Analyzer achieves an accuracy of 92.97% with the Health Watcher document as a training set. On this same training set, the false positive rate is 7.03%, i.e., an average of 2.72 *Harmony* CIs were misclassified as *Conflict*.

The classification accuracy of the tool with the Smart Home and CAS training set are, respectively, 75.94% and 63.34%. In addition, the false negative rate of the Smart Home training set is 22.15%, because 6 *Conflict* CIs were misclassified as *Harmony* in the Health Watcher data set. As seen in Table I, the Smart Home and CAS documents have less examples of the *Encryption - Performance* conflict than the Health Watcher document, and this suggests that

the amount of *Conflict* examples may have an impact on the classification accuracy.

On average, all the classification results are above the 50% baseline accuracy. However, while using the CAS document as a training set, the classification accuracy of the Health Watcher document is only 34.09%. This suggests that the size of the training set (the CAS document has only 16 CIs) can significantly influence the classification accuracy of the tool.

Validation Data	Accuracy	f_p	f_n
SH	94.20%	5.80%	0%
CAS	87.50%	12.50%	0%
Weighted Average	**92.97%**	**7.03%**	**0%**

Table II
EXPERIMENT THAT USES HW AS A TRAINING SET AND THE OTHER TWO DOCUMENTS AS A VALIDATION SET

Validation Data	Accuracy	f_p	f_n
HW	73.86%	0%	26.14%
CAS	87.50%	12.50%	0%
Weighted Average	**75.94%**	**1.90%**	**22.15%**

Table III
EXPERIMENT THAT USES SH AS A TRAINING SET AND THE OTHER TWO DOCUMENTS AS A VALIDATION SET

Validation Data	Accuracy	f_p	f_n
HW	34.09%	65.91%	0%
SH	100%	0%	0%
Weighted Average	**63.34%**	**36.66%**	**0%**

Table IV
EXPERIMENT THAT USES CAS AS A TRAINING SET AND THE OTHER TWO DOCUMENTS AS A VALIDATION SET

IV. CONCLUSIONS

Aspect-oriented requirements engineering provides an effective way to modularize and compose concerns in requirements documents. The composed concerns are an excellent starting point for analyzing conflicting dependencies. However, detecting conflicts in large aspect-oriented requirements documents that have been specified in natural-language is an error-prone and time-consuming task. Therefore, to reduce the burden on the requirements engineer, it is important to provide tools that can automate the process of identifying conflicts.

In this paper, we have presented two major contributions: a tool for automating the process of detecting conflicts within textual aspect-oriented requirements; and, an empirical evaluation of the tool with three industrial-strength requirements documents from different real-life domains. The tool is a novel application of the Naive Bayes learning method, where the problem of detecting conflicts is formulated as a text classification problem. Our empirical evaluation has shown that it is possible to detect conflicting dependencies with a high accuracy, provided that the training set has a sufficient number of examples. Thus, we see this work as a promising avenue for effort reduction in requirements conflict identification.

V. ACKNOWLEDGEMENTS

This work has been supported by a Marie Curie Fellowship from the European Commission (Grant Agreement No. PIIF-GA-2008-221016) and by the European Commission grant IST-215412 - Dynamic Variability in complex, Adaptive systems (DiVA).

REFERENCES

[1] D. Ayed, T. Genssler, Dynamic Variability in complex, Adaptive systems, Deliverable D6.1 of DiVA EC project, 2009.

[2] R. Chitchyan, A. Rashid, P. Rayson, R. Waters, 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, NY, USA, 2007.

[3] L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, 1999.

[4] R. Laney, L. Barroca, M. Jackson, B. Nuseibeh, Composing requirements using problem frames, in: RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International, IEEE Computer Society, Washington, DC, USA, 2004.

[5] T. Mitchell, Machine Learning, McGraw Hill, 1997.

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

[7] F. Mostefaoui, J. Vachon, Design-level detection of interactions in aspect-uml models using alloy, Journal of Object Technology 6 (7) (2007) 137–165.

[8] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer-Verlag New York, Inc., 2005.

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

[10] A. Sampaio, P. Greenwood, A. F. Garcia, 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, IEEE Computer Society, Washington, DC, USA, 2007.

[11] S. Soares, P. Borba, E. Laureano, Distribution and persistence as aspects, Software: Practice and Experience 36 (7) (2006) 711–759.

[12] N. Weston, R. Chitchyan, A. Rashid, A formal approach to semantic composition of aspect-oriented requirements, in: RE '08: Proceedings of the 16th International Requirements Engineering Conference, 2008.