EA-Analyzer: Automating Conflict Detection in a Large Set of Textual Aspect-Oriented Requirements

Alberto Sardinha [*]
INESC-ID and Instituto Superior Técnico,
Technical University of Lisbon
Avenida Rovisco Pais 1, Lisbon, Portugal

Ruzanna Chitchyan
Department of Computer Science
University of Leicester
Leicester, LE1 7RH, UK

Nathan Weston,
Phil Greenwood, Awais Rashid
Computing Department
Lancaster University
Lancaster, LA1 4WA, 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. A composition definition explicitly represents interdependencies and interactions between concerns. Subsequent analysis of such compositions helps to reveal conflicting dependencies that need to be resolved in requirements. However, detecting conflicts in a large set of textual aspect-oriented requirements is a difficult task as a large number of explicitly defined interdependencies need to be analyzed. 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.

[*]Corresponding author. Email Address: `jose.alberto.sardinha@ist.utl.pt`; Telephone: +351-21-4233291; Fax: +351-21-8417789.

We present an empirical evaluation of the tool with three industrial-strength requirements documents from different domains and a fourth academic case study used as a de facto benchmark in several areas of the aspect-oriented community. This evaluation shows that the tool achieves up to 93.90% accuracy regardless of the documents chosen as the training and validation sets.

1 Introduction

Aspect-Oriented Requirements Engineering (AORE) [2] [4] [37] 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. A concern that intersects with other concerns is called a crosscutting concern or an aspect. For instance, a set of security requirements that crosscut other requirements is called a security aspect in AORE.

A composition is used to explicitly represent and analyze the interdependencies between concerns. Compositions are also utilized for detecting potential conflicts between concerns before an architecture structure is derived. However, detecting conflicts in a large set of textual Aspect-Oriented (AO) requirements is a difficult task as a significant number of explicitly defined interdependencies need to be analyzed.

Presently, there are three streams of research on conflict detection in AO textual requirements: (i) formalization of requirements and compositions; (ii) model-based representations and analysis; and (iii) stakeholder priority-based analysis. Yet, each of these research directions has its drawbacks. The formalization-based work [23] [31] [49] for detecting conflicts requires transformation of textual requirements into specific formal representations, costing substantial time and effort. Conversion of text and compositions into models for analysis [27] [5] often leads to losing relevant information. Finally, the stakeholder priority-based work [7] is both effort intensive, and relies on subjective judgement of each stakeholder. As a result, conflict detection in large textual specifications is an error-prone and time-consuming task that puts a burden on the requirements engineer, and is still often performed fully manually via visual inspection.

This paper presents EA-Analyzer, the first automated tool for identifying conflicts in textual AO requirements. The tool operates on annotated natural language text and compositions defined using these annotations [11]. These annotations do not alter or reduce the textual requirements, but only decorate text with syntactic and semantic linguistics tags [39]. A Bayesian learning method, called Naive Bayes [28], is utilized by EA-Analyzer to learn the nature of the composed concerns and to detect conflicts within the composition specifications. We evaluated the tool with three industrial requirements documents and a well established academic case study: a Web-based application that manages health-related complaints [47], a home automation system [36], a customer relationship management application [3], and a crisis management system [22]. The results show that EA-Analyzer achieves very high accuracy results, up to 93.90%, regardless of the documents chosen, as long as the training documents have a sufficient number of training examples.

The key contributions of this paper are twofold. First, we demonstrate the technical feasibility of automated conflict detection in textual AO requirements. Here we also describe in detail the learning method used by EA-Analyzer to automate the process of detecting conflicts. The learning method in EA-Analyzer is a novel application of the Naive Bayes classifier, in which the problem of detecting conflicts is formulated as a text classification problem. Second, we show that it is possible to detect conflicts with a high accuracy in textual AO requirements, provided that the training set has a sufficient number of examples. In order to confirm the second contribution, we present an empirical evaluation of the tool with four requirements documents, where three documents originate from industrial organizations from different domains and the fourth document is a well established case-study widely used by many AO researcherss to evaluate various modeling techniques.

This paper is organized as follows. Section 2 provides some background about conflict detection techniques in AORE. Section 3 describes the process for detecting conflicts in a large set of textual AO requirements utilizing the EA-Analyzer tool, one of the key contributions of this paper. Section 4 presents the empirical evaluation of the tool, another key contribution of this paper. Section 5 provides a brief overview of relevant work in natural language processing approaches for requirements engineering (RE), conflict detection in RE, and machine learning approaches for text classification. Finally, the conclusions are presented in Section 6.

2 Background

AORE techniques enable the early identification of candidate crosscutting concerns within problem domains. These strategies allow requirements engineers to specify how requirements interact with one another. AORE approaches provide significant advantages for reasoning about requirements, as mutual influences and tradeoffs can be identified before an architecture structure is derived. In addition, the transition to an AO architecture can be eased by the explicit recognition of early aspects within the domain.

However, AORE also brings with it a significant challenge - namely, the accurate detection of *conflicts* between requirements. Moreover, detecting conflicts in requirements that interact with one another can become very time consuming in a large set of textual requirements. The issue of detecting conflicts has received a great deal of research attention within the AO community when the conflict is expressed at the implementation level; but research at the requirements level is much less mature. In this section, we perform a brief survey of existing AORE approaches that support conflict detection, before explaining the approach we take and presenting our conflict detection tool in subsequent sections.

2.1 Formalization-Based approaches

Many current conflict detection approaches require some formal specification of requirements in order to support this activity. In other words, they require precise expression of the properties of requirements in order to determine whether compositions of requirements invalidate these properties.

Laney et. al. [23] introduce Composition Frames, which model the semantics of requirements (in the form of Problem Frames) being composed with one another. The requirements of the composition - that is, the formal properties of its satisfaction - can be validated against the state machine expressed in the Composition Frame, and conflicts detected. The validity of the conflict detection thus depends on the sound construction of the Problem Frames and their compositions.

Mostefaoui and Vachon [31] consider AO models specified in Aspect-UML, including 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 to check the validity of assertions over a model. The Aspect-UML model of an AO system can be checked for aspects introducing properties that invalidate other aspect assumptions, thus identifying conflicts.

Similarly, Weston et. al. [49] present a conflict detection technique based on the Requirements Description Language [11]. The basis of this approach is the transformation of compositions into temporal logic formulae using a catalogue of formalizations of natural language operators. The semantics of the compositions can thus be compared with others to identify temporal comflicts between requirements.

The major disadvantage of these approaches is the substantial time and effort needed to transform requirements into the required formal representations. In addition, any errors introduced during the formalization process can lead to an inaccurate representation of the conflicts present in the requirements.

2.2 Model-based approaches

As an alternative to the formalization-based approaches, many approaches take a design-level view of conflict or interaction detection; that is, they require requirements to be (at least initially) structured into specific models before conflicts can be detected.

Mehner et. al. [27] model requirements as use-cases in UML, and the crosscutting concerns are modeled using activities that refine those use-cases. The approach then translates these UML diagrams into type graphs, and activities are modeled as graph transformations. Applying these graph transformations sequentially can reveal conflicts between requirements. A similar technique based on statechart weaving on UML models was proposed in [45].

In a similar vein, Barais et. al. [5] adapt the Theme/UML [4] approach to formally model compositions between base and aspect concerns. Certain forms of conflict based on global properties, such as visibility and kind, can be discerned and automatically resolved. A similar technique for class diagrams was presented in [40].

The disadvantage of such approaches are twofold. First, the modelling process adds an extra step to the conflict detection process, which may require additional time and effort. Second, the structuring of requirements into models may be a lossy technique, so the information encoded in the requirements, including potential conflicts, may be hidden from the engineer when analysing the interactions.

2.3 Stakeholder priority-based approaches

Other approaches to conflict detection take a higher-level view of interactions between concerns based on their relative importance to stakeholders. If interactions can be identified using a technique such as ARCADE [37], the stakeholders can then determine whether such compositions are positive, negative or neutral from their point of view, and refine the requirements accordingly [37]. Alternatively, stakeholders state their preferred non-functional requirements up-front; mathematical reasoning techniques can then be applied to assist conflict resolution [7].

These approaches are very useful, because stakeholder engagement is a strong element for resolving conflicts. Tools for detecting conflicts, such as the one presented in this paper, can be used to ascertain the existence of conflicting dependencies in large requirements document, helping the stakeholders select the conflicts that need to be resolved.

3 Detecting Conflicts in a Large Set of Textual Aspect-Oriented Requirements

This section describes the process used by EA-Analyzer to detect conflicts in a large set of textual AO requirements. The process starts with the annotation of a requirements document written in natural language. The annotation is conducted semi-automatically using a tool called EA-Miner [42]. EA-Miner processes the textual requirements and a requirements analyst specifies compositions to reflect the relationships between requirements of a given domain. The annotated text along with the compositions is then used by EA-Analyzer to detect conflicts within the requirements document. Below a running example is used to illustrate the process.

3.1 Running Example: Web-based Information System

Our running example - the Health Watcher (HW) system [47] - is a typical Web-based information system. The system is used by citizens to register health related complaints. The requirements document of HW is rich in both functional and non-functional requirements with many crosscutting concerns, such as security, performance and distribution. Figure 1 presents the two non-functional requirements in the HW specification that we use to illustrate the process of detecting conflicts in a AO textual requirements. The reasons for selecting these two requirements are the following: (i) The two requirements crosscut other requirements in the specification (e.g., the requirements in Figure 2(c)); and, (ii) The two non-functional requirements are well-known examples of potential conflicts between *Security Mechanism*, such as *Encryption*, and *Performance* [43].

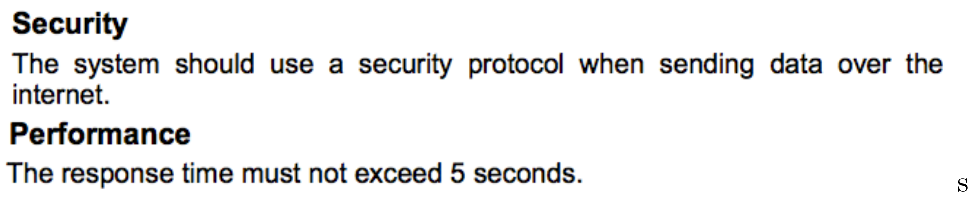

Security
The system should use a security protocol when sending data over the internet.
Performance
The response time must not exceed 5 seconds.
s

Figure 1: Security and Performance Requirements in the HW Specification

3.2 Annotating Textual Requirements with the Requirements Description Language

The annotations of the Requirements Description Language (RDL) [11] utilize the richness of the natural language for expressing the dependencies and interactions between various groups of requirements (such as viewpoints, use-cases, etc.) in large textual specifications. Related sets of requirements are grouped into a *concern* - or module of interest - in RDL. Figure 2(a) shows an example of the Security requirement in Figure 1 that has been annotated with the RDL tags. The process of adding the RDL tags to the text of requirement is performed by EA-Miner [42].

RDL tags express dependencies and interactions between requirements. These tags allow the analyst to define domain relationships (via compositions) using only natural language text. For example, RDL compositions can mandate that certain requirements must precede another, such as an *Encryption* requirement must be satisfied before any requirement that sends data. Another example could specify that a particular action is carried out by an *actor* or on an *element* whenever a certain condition is met (e.g., *prohibit download* of files whenever the system is used from an untrusted workstation). The details of RDL tags structure, composition definitions, and annotation automation are presented in [11]. The annotated requirements and compositions are then used by EA-Analyzer to detect conflicts, this process is presented in Section 3.3.

Since interactions and dependencies expressed in compositions tend to be relevant for actions and requirements that are often located across modules, a set of interaction points will have to be identified for their representation. For instance, sending data is an action that appears in many concerns in the HW system, such as login, complaint updating, and new employee registration. RDL supports direct referencing to such interesting points (also called *pointcuts* in Aspect-Oriented Software Development terminology) through the natural language itself. The direct referencing in RDL differs from the

```
<Concern name="Security">
    <Requirement id="1">The
      <Subject>system</Subject>
      <Degree type="modal" semantics="obligation"
      level="high">should</Degree>
      <Relationship type="General_Action" semantics="Compare"> use
      </Relationship> a
      <Object>security protocol</Object>when
      <Relationship type="Move" semantics="Set_in_Motion"> sending
      </Relationship>
      <Object>data</Object> over the internet.
    </Requirement>
…
</Concern>
```

(a) The RDL annotated extract of a requirement of the Security concern

```
<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>
```

(b) Composition enforcing use of a security protocol (i.e., encryption protocol) when sending or 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."
…

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

Figure 2: An Example of the RDL from the HW System

current established practice of using extraneous syntactic scaffolding, such as manually finding and providing name or number-based IDs to any part of text where such points of interest are located (e.g., use-case step numbers).

The referencing mechanism of the RDL is based on the natural language itself and relies on its clearly defined set of syntactic rules and semantic elements, precise enough to support definition of a flexible composition mechanism for requirements analysis. The RDL annotations are added as XML tags to the syntactic elements of the natural language, exploiting the designated semantic role each syntactic element has. An example is shown in Figure 2(a): the natural language requirements are annotated with additional information on their grammatical and semantic properties. Grammatical properties are related to the grammatical functions of the words, the main ones being:

- Subject: the entity that performs the main action of the sentence, or its main theme (e.g., the **system** in Figure 2(a));
- Relationship: the main activity (e.g., **use** in Figure 2(a)) or property of the sentence;
- Object: the entities most affected by the activity in the sentence, or with respect to which the activity is realized (e.g., **data** in Figure 2(a)).

The semantic properties are related to grouping of words on the basis of synonyms (e.g., send and transmit as shown in Figure 2(c)) or type. For instance, verb types are based on the notion of a set of participatory roles engaged in the given activities (e.g., both "Citizen sent the complaint" and "Susan threw the ball" imply that someone (Citizen or Susan) playing the Causer role puts into motion (send, throw) some Moving Thing role (complaint, ball)). Such grammatical and semantic annotations of the requirements text in the RDL are provided via a general purpose Natural Language Processing (NLP) tool, Wmatrix [39], that is used by EA-Miner [42] to generate the annotated RDL text.

3.2.1 Compositions

As shown in Figure 2(b), an RDL composition consists of three parts: *Constraint*, *Base*, and *Outcome*. Each of these parts has a semantic query (i.e., pointcut) expressed in terms of the natural language words and their properties. These queries select requirements (i.e., joinpoints) from all across the specification document without reference to any structural information, such as a requirement ID or string-based name matching.

For instance, the query *relationship="send" or relationship="receive"* in the Base element of the composition in Figure 2(b) will select the requirement *The system should use a security protocol when* ***sending*** *data over internet* in Figure 2(a), where the *send* verb of the query will match the *sending* verb of the requirement. Similarly, if there were any other requirements either directly or via synonymy referring to *send*, they would also be selected by the query in the Base element of Figure 2(b) (an example is presented in Figure 2(c)). For each composition, a Constraint query selects some requirements which crosscut the requirements selected by the Base element's query. The Outcome element may select some requirements which should be checked as post-conditions (as in Figure 2(b)), though in some cases the outcome may have an empty query (if no post-condition needs to be checked). The details of the RDL are presented in [11, 9].

3.3 Detecting Conflicts with EA-Analyzer

EA-Analyzer detects conflicts within a textual AO 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 [28]. The tool operates on RDL in both the learning phase and the conflict detection phase. In other words, the compositions and annotated requirements are used to train the tool in the first phase, and then utilizes the composed concerns from the RDL to decide whether or not they have a conflicting dependency among requirements.

Recall that compositions in AORE are utilized to explicitly represent and analyze the interdependencies between requirements. In this context, a conflict occurs when a crosscutting (functional or non-functional) requirement has a negative contribution with another (functional or non-functional) crosscutting requirement on 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.

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 3.3.1); (ii) Generate training examples for the learning method by labeling the compositional intersections (Section 3.3.2); and, (iii) Train the classifier based on the examples generated in step (ii) (Section 3.3.3).

3.3.1 Identifying 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. This section describes the algorithm used to identify compositional intersections, which is a modified version of the algorithm in [30].

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 AO 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.

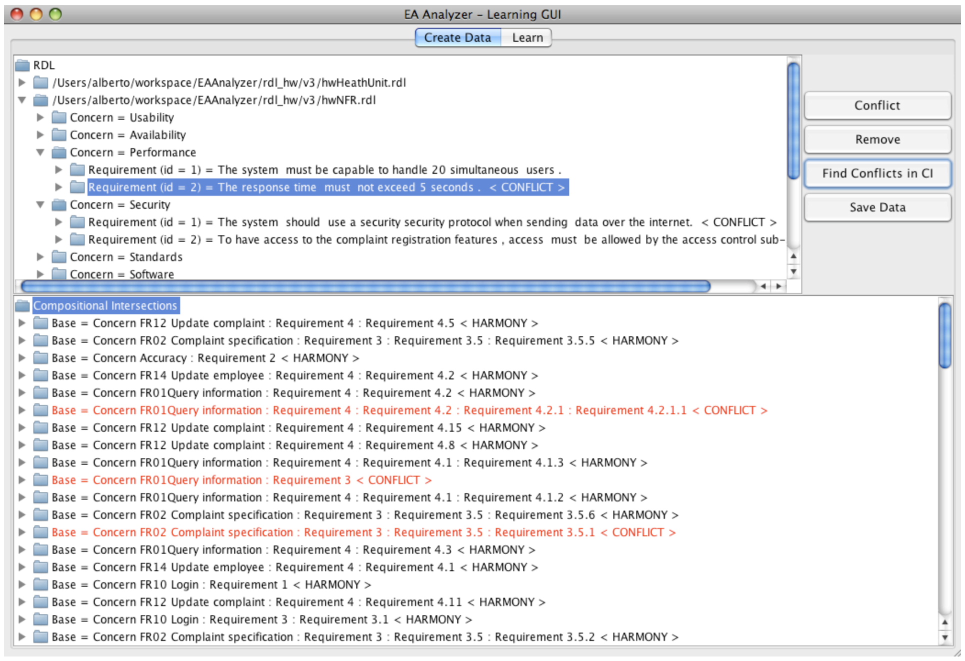
$$CI_{i,j} = \bigcup_{k \in Sc_{i,j}} Cr_k \tag{1}$$

A compositional intersection is the union of all the constraint requirements that crosscut the same base requirement. For example, the composition in Figure 2(b) selects the constraint requirement $R_{1,1}$ = "The system should use a security protocol when sending data over the internet" to crosscut the base requirements in Figure 2(c) ($R_{2,1}$ = "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"). The HW system also has a composition that selects a 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.

3.3.2 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 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 a user interface (UI) to the tool that helps a user label each compositional intersection and save the training data for the learning process. Figure 3 presents the UI in EA-Analyzer that helps a human annotator label the compositional intersections. In the UI, the human annotator is required only to select the conflicting requirements from the top list, and the



```
RDL
▶ /Users/alberto/workspace/EAAnalyzer/rdl_hw/v3/hwHeathUnit.rdl
▼ /Users/alberto/workspace/EAAnalyzer/rdl_hw/v3/hwNFR.rdl
    ▶ Concern = Usability
    ▶ Concern = Availability
    ▼ Concern = Performance
        ▶ Requirement (id = 1) = The system  must be capable to handle 20 simultaneous  users .
        ▶ Requirement (id = 2) = The response time  must  not exceed 5 seconds .  < CONFLICT >
    ▼ Concern = Security
        ▶ Requirement (id = 1) = The system  should  use a security security protocol when sending  data over the internet.  < CONFLICT >
        ▶ Requirement (id = 2) = To have access to the complaint registration features , access  must  be allowed by the access control sub-
    ▶ Concern = Standards
    ▶ Concern = Software

Compositional Intersections
▶ Base = Concern FR12 Update complaint : Requirement 4 : Requirement 4.5 < HARMONY >
▶ Base = Concern FR02 Complaint specification : Requirement 3 : Requirement 3.5 : Requirement 3.5.5 < HARMONY >
▶ Base = Concern Accuracy : Requirement 2 < HARMONY >
▶ Base = Concern FR14 Update employee : Requirement 4 : Requirement 4.2 < HARMONY >
▶ Base = Concern FR01Query information : Requirement 4 : Requirement 4.2 < HARMONY >
▶ Base = Concern FR01Query information : Requirement 4 : Requirement 4.2 : Requirement 4.2.1 : Requirement 4.2.1.1 < CONFLICT >
▶ Base = Concern FR12 Update complaint : Requirement 4 : Requirement 4.15 < HARMONY >
▶ Base = Concern FR12 Update complaint : Requirement 4 : Requirement 4.8 < HARMONY >
▶ Base = Concern FR01Query information : Requirement 4 : Requirement 4.1 : Requirement 4.1.3 < HARMONY >
▶ Base = Concern FR01Query information : Requirement 3 < CONFLICT >
▶ Base = Concern FR01Query information : Requirement 4 : Requirement 4.1 : Requirement 4.1.2 < HARMONY >
▶ Base = Concern FR02 Complaint specification : Requirement 3 : Requirement 3.5 : Requirement 3.5.6 < HARMONY >
▶ Base = Concern FR02 Complaint specification : Requirement 3 : Requirement 3.5 : Requirement 3.5.1 < CONFLICT >
▶ Base = Concern FR01Query information : Requirement 4 : Requirement 4.3 < HARMONY >
▶ Base = Concern FR14 Update employee : Requirement 4 : Requirement 4.1 < HARMONY >
▶ Base = Concern FR10 Login : Requirement 1 < HARMONY >
▶ Base = Concern FR12 Update complaint : Requirement 4 : Requirement 4.11 < HARMONY >
▶ Base = Concern FR10 Login : Requirement 3 : Requirement 3.1 < HARMONY >
▶ Base = Concern FR02 Complaint specification : Requirement 3 : Requirement 3.5 : Requirement 3.5.2 < HARMONY >
```

Figure 3: The User Interface for Generating Training Examples

tool automatically labels each compositional intersection by using a brute force procedure that labels each occurrence of the conflicting dependency.

The example in Section 3.3.1 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 $R_{2,3}$. Recall that the compositional intersection in Section 3.3.1 is a well-known example of a potential conflict between *Security Protocol*, such as *Encryption*, and *Performance* [43]. In EA-Analyzer, the user is required only to select these requirements, and the tool is made responsible for labeling all the compositional intersections (i.e., either *Conflict*, if both requirements crosscut the same base, or *Harmony*, otherwise) in the requirements document .

3.3.3 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 [28] classifier, which is an effective approach to the problem of learning to classify text [26, 28, 16].

In EA-Analyzer's text classification problem, the input vector of features is extracted from the text of the requirements within a compositional intersection, 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 compositional intersection (see Appendix A for detailed explanation).

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

where $< a_1, a_2, ..., a_n >$ is the input vector of features and V is a finite set of classes. The aim of the Naive Bayes 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 in Figure 2(a) 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 the compositional intersection, we instantiate Equation 2 as follows:

$$\begin{aligned} v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j)P(a_1|v_j) \\ ...P(a_i = \text{``}security\ protocol\text{''}|v_j) \\ P(a_j = \text{``}data\text{''}|v_j) \\ ...P(a_n|v_j) \end{aligned}$$

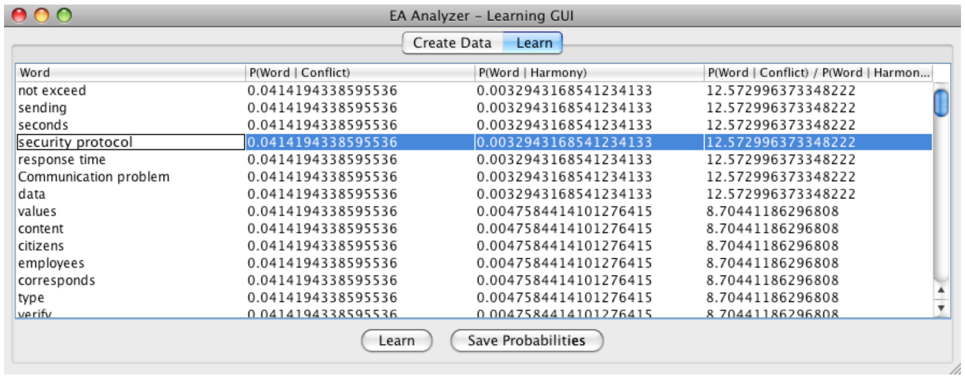
The extraction of words from the requirements is a pre-processing stage in machine learning called *feature extraction* [6]. The aim is to pre-process an original input into a new space of features, in which the classification problem is easier to solve. For instance, in the example above, the words "security protocol" and "data" are selected as features for the Naive Bayes classifier, while other words such as "a" and "over the internet" have not been selected. The current version of EA-Analyzer relies on the RDL annotation process as a method for extracting features.

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.

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

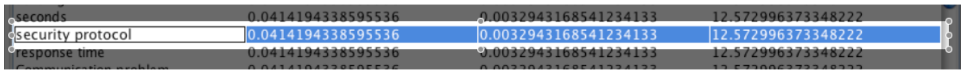
$$P(a_i|v_j) = \frac{n_k + \theta}{n + \theta.|Vocabulary|} \tag{4}$$

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.



Word	P(Word | Conflict)	P(Word | Harmony)	P(Word | Conflict) / P(Word | Harmon...
not exceed	0.0414194338595536	0.0032943168541234133	12.572996373348222
sending	0.0414194338595536	0.0032943168541234133	12.572996373348222
seconds	0.0414194338595536	0.0032943168541234133	12.572996373348222
security protocol	0.0414194338595536	0.0032943168541234133	12.572996373348222
response time	0.0414194338595536	0.0032943168541234133	12.572996373348222
Communication problem	0.0414194338595536	0.0032943168541234133	12.572996373348222
data	0.0414194338595536	0.0032943168541234133	12.572996373348222
values	0.0414194338595536	0.0047584414101276415	8.70441186296808
content	0.0414194338595536	0.0047584414101276415	8.70441186296808
citizens	0.0414194338595536	0.0047584414101276415	8.70441186296808
employees	0.0414194338595536	0.0047584414101276415	8.70441186296808
corresponds	0.0414194338595536	0.0047584414101276415	8.70441186296808
type	0.0414194338595536	0.0047584414101276415	8.70441186296808
verify	0.0414194338595536	0.0047584414101276415	8.70441186296808

(a) The Estimated Probabilities



seconds	0.0414194338595536	0.0032943168541234133	12.572996373348222
security protocol	0.0414194338595536	0.0032943168541234133	12.572996373348222
response time	0.0414194338595536	0.0032943168541234133	12.572996373348222

(b) The Estimated Probability of "security protocol"

Figure 4: The User Interface for Analyzing the Estimated Probabilities

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 4 shows a table with the words sorted by the fraction $\frac{P(a_i|Conflict)}{P(a_i|Harmony)}$, this is a useful method for analyzing the most probable words in the *conflict* class. For example, the estimated probabilities for "security protocol" in Figure 4 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 $Conflict$.

3.3.4 Advantages and Disadvantes of the Learning Method in EA-Analyzer

The Bayesian learning technique in EA-Analyzer leads to a bag of words model (BoW) [25]. The BoW is a method in NLP that models text as an unordered collection of independent words represented in a term-frequency vector, disregarding grammar[1] and even word order.

In EA-Analyzer's BoW, one can imagine there are 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 in Section 3.3.1. 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 "security protocol" and "sending" much more frequently. On the other hand, the second bag will contain more words related to the other requirements. Thus, if the text in the requirements of a new compositional intersection has more words that come from the first bag than the second bag, then it will be classified as a conflict.

The main advantages of the learning method in EA-Analyzer are twofold. Firstly, the learning method only requires a small amount of training data to estimate the parameters in the Naive Bayes classifier [28]. Secondly, this learning method can be easily trained on a per-user 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. The 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. The same technique is used in EA-Analyzer in the feature extraction stage as described in Section 3.3.3.

We must also note that the binary classification of a relationship 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. EA-Analyzer will always pinpoint the potential presence of such conflicts. It is then up to the requirements analyst to consider if a

[1]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.

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

4.1 Study Configuration

In order to conduct an effective evaluation of the EA-Analyzer tool, careful consideration had to be given to the study configuration. Firstly, a series of textual requirement documents had to be selected that could be analyzed for conflicts. The documents were selected based on their suitability for such a study, with selection criteria including: domain, requirement type, complexity and use in previous studies. Four documents were selected for this study, where 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 include:

- *HealthWatcher* [47] is a web-based health support system which the public can use to register health-related complaints and query disease and symptom information.
- *SmartHome* [36] is an embedded system which provides functionality to control various sensors and actuators around the home (i.e., lights, blinds, heating, etc.).
- *CAS* [3] 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* [22] is a crisis management system for emergency situations (e.g., natural disasters, accidents, terrorist attacks).

The documentation for each of these applications contains 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 work has been undertaken to determine whether these requirements do actually conflict in each of these documents.

In order to process the documents with EA-Analyzer, each of them had to be pre-processed using the RDL (see Section 3). In a previous study [10], the HW requirements had already been re-structured using the RDL. Additionally, this study required evolving the HW RDL specification by adding new requirements. Thus, we were able to evaluate EA-Analyzer's ability to identify conflicts in evolving documents (Section 4.1.2). To prepare the other three documents using the RDL, two researchers were selected who had an in depth knowledge of the SmartHome, the CRM and the Crisis Management systems. Furthermore, both of these researchers had previous experience of using the RDL to structure requirements documents. The process of creating the RDL specification was supported by EA-Miner [42]. The output from EA-Miner provided the researchers with the annotated sets of potential requirements and concerns. However, it was the responsibility of the researchers to first determine whether the output was correct and then create the appropriate compositions between the requirements of concerns. Once RDL compositions were created, the texts and compositions were provided to EA-Analyzer for processing.

	HW1	HW2	SH	CAS	CM
Words in RDL	1521	1764	4699	1053	5961
Num. of Compositions	15	17	9	5	8
Num. of CI	77	89	71	16	43
Num. of Enc-Acc	0	14	4	0	16
Num. of Enc-Perf	23	23	5	3	16
Num. of FF-Avail	41	42	0	0	0
Num. of Ind-Sp	0	0	3	0	0
Num. of Ver-Conf	0	0	0	2	0

Table 1: Characteristics of the Requirements Documents

Table 1 shows some characteristics of the four documents selected for this study: two versions of HW (HW1 and HW2), SmartHome (SH), CAS and Crisis Management System (CM). The characteristics present two different dimensions of the documents: (i) the size of the documents, by showing the number of words, compositions and compositional intersections (CI); and (ii) The types of conflict, by showing the number of compositional intersections that have a specific type of well-known conflicting dependency (e.g. between Encryption and Accuracy). A number of these well-known conflicts between non-functional requirements (NFR) have been previously identified with each system used in our evaluation containing at least two different types of these conflicts:

- *Encryption - Accuracy (Enc-Acc)* [12] - Encryption reduces the chance of direct examination, a method for operationalizing Accuracy.
- *Encryption - Performance (Enc-Perf)* [43] - Introducing encryption into a system reduces its responsiveness.
- *Fix Fault - Availability (FF-Avail)* [19] - Fixing faults increases the downtime of a system, which reduces its availability.
- *Indexing - Space (Ind-Sp)* [12] - Indexing has a negative impact on Space performance.
- *Verification - Confidentiality (Ver-Conf)* [12] - Verification, an operationalizing method of Accuracy, may hurt Confidentiality to ensure access to the data if it has to be verified.

We have the following set of hypotheses for this evaluation. The first null and alternative hypotheses relate to the accuracy of the tools, while the second null and alternative hypotheses relate to the effectiveness of the type of training data used:

- Null hypothesis 1 - $H1_0$: A low classification accuracy will be achieved when testing the tool using training data from the same domain.
- Alternative hypothesis 1 - $H1_1$: A high classification accuracy will be achieved when testing using training data from the same domain.
- Null hypothesis 2 - $H2_0$: The performance of the tool will remain the same regardless of the training data selected for the learning method.
- Alternative hypothesis 2 - $H2_1$: The performance of the tool will differ depending on the training data selected for the learning method.

We conducted three experiments to test the above hypotheses. First, we tested hypothesis H1 by performing cross-validation within each of the documents (HW2, SH, CAS and CM), where each example is tested in turn (i.e., each compositional intersection in turn) as the validation data and the remaining examples as the training data (Section 4.1.1). The technique for testing H1 is known as leave-one-out cross-validation [6] and is commonly used in machine learning to estimate the generalization error of a learning technique on a data set.

The next set of experiments test hypothesis H2, whilst providing further evidence for hypothesis H1. The first involved ascertaining the accuracy of EA-Analyzer when attempting to find *new* conflicts in *different* versions of the same requirements document (Section 4.1.2). Then, we assessed the ability of EA-Analyzer to detect conflicts using training data gathered from different domains (Section 4.1.3), where the training data was gathered from each of the requirements documents in turn and applied to the remaining documents in order to identify the conflicts.

4.1.1 Cross-Validation

In a data set with N examples, each round of the leave-one-out cross-validation procedure [6] utilizes $N-1$ examples to train a machine learning technique and then validates it on the remaining example. In order to compute the accuracy of a machine learning technique, the validation results are averaged over the rounds.

For instance, a leave-one-out cross-validation procedure for a data set with three examples $\{E_1, E_2, E_3\}$ would run as follows:

- Run 1: Train the machine learning technique with $\{E_2, E_3\}$ and assess the performance with $\{E_1\}$. This generates performance score S_1.
- Run 2: Train the machine learning technique with $\{E_1, E_3\}$ and assess the performance with $\{E_2\}$. This generates performance score S_2.
- Run 3: Train the machine learning technique with $\{E_1, E_2\}$ and assess the performance with $\{E_3\}$. This generates performance score S_3.

The accuracy of the leave-one-out cross-validation technique is then computed as follows:

$$S = \frac{S_1 + S_2 + S_3}{3}$$

Tables 2 - 5 present the results of the leave-one-out cross-validation technique (now referred to as cross-validation) with a 95% confidence interval (i.e., we are 95% confident that the interval contains the true population mean). Each document has at least two types of NFR conflicts, and were created using natural language from different personnel. All the results are compared to a baseline accuracy of 50% as randomly assigned classes should yield an approximate 50% accuracy. Overall, the results show that the learning method in EA-Analyzer was able to classify the compositional intersections as *Conflict* or *Harmony* with an accuracy of 98.35% (average of the results in Tables 2 - 5). Moreover, all the results are above the 50% baseline accuracy.

Conflict	C.V. ±95% Conf.Int.	f_p	f_n
Enc-Acc	98.86% ± 2.23%	0%	1.14%
Enc-Perf	100%	0%	0%
FF-Avail	100%	0%	0%

Table 2: Cross-Validation - HW2

Conflict	C.V. ±95% Conf.Int.	f_p	f_n
Enc-Acc	100%	0%	0%
Enc-Perf	100%	0%	0%
Ind-Sp	100%	0%	0%

Table 3: Cross-Validation - Smart Home

Conflict	C.V. ±95% Conf.Int.	f_p	f_n
Enc-Perf	84.62% ± 20.41%	15.38%	0%
Ver-Conf	100%	0%	0%

Table 4: Cross-Validation - CAS

Conflict	C.V. ±95% Conf.Int.	f_p	f_n
Enc-Acc	100%	0%	0%
Enc-Perf	100%	0%	0%

Table 5: Cross-Validation - CM

It is important to note that misclassifying a *Harmony* compositional intersection may have a different impact to misclassifying a *Conflict* compositional intersection. Thus, we also report the rate of false positives (f_p) and false negatives (f_n); false positive rate is the proportion of *Harmony* compositional intersections classified as *Conflict*, and the false negative rate is the proportion of *Conflict* compositional intersections classified as *Har-*

mony. In the CAS document, the Encryption - Performance conflict presents a false positive rate of 15.38%, which means that 2 *Harmony* compositional intersections were misclassified as *Conflict.*

The cross-validation results in Tables 2 - 5 show that EA-Analyzer can learn to detect conflicts with a high accuracy, which leads us to reject $H1_0$ and accept $H1_1$. The results also show that documents with a small number of compositional intersections, such as the CAS document with 16 (see Table 1), may present a lower accuracy.

4.1.2 Testing a Requirements Document that Evolved Over Time

This experiment involves evaluating the classification accuracy of EA-Analyzer with a document that has evolved over time. In this evaluation, we selected the first version of the HW document [47] to be our training set, and used the evolved document in [10] to test the classification accuracy of the tool. It is important to note that the first version of the document has only two NFR conflict types (see Table 1), so the tool can detect only these conflict types in the evolved version.

Table 6 shows the results of this experiment. On average, the classification accuracy is 92.61% and the false positive rate is 7.39%, which represents 4 *Harmony* compositional intersections (out of 89) being misclassified as *Conflict.* These results also show that the tool can achieve a high classification accuracy in this different experimental setting and provides further evidence to accept hypothesis $H1_1$.

Conflict	Accuracy	f_p	f_n
Enc-Perf	94.45%	4.55%	0%
FF-Avail	89.77%	10.23%	0%
Average	**92.61%**	**7.39%**	**0%**

Table 6: Experiment that uses HW1 as a training set and HW2 as a validation set

4.1.3 Using a Requirements Document to Learn to Detect Conflicts in Other Documents

In this experiment, we used each requirements document (HW2, SH, CAS and CM) as a training set, and evaluated the classification accuracy of the tool with the other three documents. This is the most challenging test for

EA-Analyzer, because it tests 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 3.3.3, the words in the training set (i.e., the words in the document) are used as features to the classifier. However, some of the words appear only in one document (e.g., the HW2 document uses the term "security protocol" in a security requirement, while the other documents use the term "encryption" for an equivalent security requirement), which can significantly influence the classification accuracy of the tool. Therefore, to address this issue, we provide a synonym list to the tool so that it can match words that have the same meaning across multiple documents.

Tables 7 - 10 present the classification accuracy of the tool with the four different training sets. In this experiment, we used the 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 (see Table 1). In Table 7, EA-Analyzer achieves an accuracy of 93.90% with the HW2 document as a training set. On the same training set, the false positive rate is 6.10%, i.e., an average of 4.02 *Harmony* compositional intersections were misclassified as *Conflict*. The experiment that uses the Crisis Management document as a training set yields the same results as the HW2 document.

The classification accuracy of the tool with the Smart Home and CAS training set are 75.94% and 63.34% respectively. In addition, the false negative rate of the Smart Home training set is 22.15%, because 6 *Conflict* compositional intersections were misclassified as *Harmony* in the HW2 data set. As seen in Table 1, the Smart Home and CAS documents have fewer examples of the *Encryption - Performance* conflict than the HW2 document, which suggests that the amount of *Conflict* examples may have an impact on the classification accuracy.

The weighted averages of this experiment are all above the 50% baseline accuracy. However, the classification accuracy of the HW2 document in Table 9 is 34.09%, which suggests that the size of the training set (the CAS document has only 16 compositional intersections) can significantly influence the classification accuracy of the tool and so causes us to reject hypothesis $H2_0$ and accept $H2_1$.

4.2 Threats to Validity

When conducting a study of this nature a variety of threats exist which can invalidate the collected results. The purpose of this section is to identify

Validation Data	Accuracy	f_p	f_n
SH	94.20%	5.80%	0%
CAS	87.50%	12.50%	0%
CM	100%	0%	0%
Weighted Average	**93.90%**	**6.10%**	**0%**

Table 7: Experiment that uses HW2 as a training set and the other two documents as a validation set

Validation Data	Accuracy	f_p	f_n
HW2	88.64%	0%	11.36%
CAS	87.50%	12.50%	0%
CM	100%	0%	0%
Weighted Average	**92.05%**	**4.17%**	**3.79%**

Table 8: Experiment that uses SH as a training set and the other two documents as a validation set

Validation Data	Accuracy	f_p	f_n
HW2	34.09%	65.91%	0%
SH	100%	0%	0%
CM	11.43%	88.57	0%
Weighted Average	**48.51%**	**51.49%**	**0%**

Table 9: Experiment that uses CAS as a training set and the other two documents as a validation set

Validation Data	Accuracy	f_p	f_n
HW2	100%	0%	0%
SH	94.20%	5.80%	0%
CAS	87.50%	12.5%	0%
Weighted Average	**93.90%**	**6.10%**	**0%**

Table 10: Experiment that uses CM as a training set and the other two documents as a validation set

some of the potential threats to validity and outline the steps undertaken to limit their effect. It is inevitable that threats to validity can never be eliminated completely, however, assurances can be given that any affect is minimal and considered.

The most significant threat to validity of this experiment is the quality of the RDL specification created by the participants involved in the study. It is possible that the compositions which they have created are not a true representation of the application. The creation of such compositions could cause conflicts to be introduced that do not actually exist or conflicts omitted which do exist. Moreover, this is a threat of the EA-Analyzer approach in general, however, tool support is available to improve and aid developers when constructing the RDL specifications. To ensure the quality of the RDL for this study, the participants chosen to create the necessary RDL and verify the results had an appropriate degree of proficiency in both the RDL technique and the domains of the requirements documents. In some cases, the authors of the original requirements documents were consulted.

Although a range of requirement documents from different domains have been analyzed in this study, further analysis is needed to validate the wider generality of the results collected here. For instance, documents from a larger number of domains and with different sets of conflicting requirements could help to further validate the ability of EA-Analyzer to detect conflicts using existing training data. Similarly different documents are needed in terms of size, style (i.e., use of language, document type, etc.) and stages of maturity (i.e., initial version of the document up to the final version) to verify the applicability of EA-Analyzer in different conditions. Despite not fulfilling all of this criteria, this study has shown that EA-Analyzer does have promise and is a useful tool for detecting conflicts in textual requirements.

4.3 Discussion of Results

The findings in this section show that EA-Anlyzer can achieve good results in detecting conflicts in textual AO requirements. However, the outcomes are not satisfactory if the requirements documents have only a small number of examples of conflicts analyzed. As discussed previously, this has negatively affected the findings of the CAS application producing noticeably worse results due to its small number of compositions and intersections. The lack of examples makes it difficult to train EA-Analyzer for the CAS application causing inaccuracies in the results. This type of scenario is typical of all machine learning approaches where the lack of training data will inevitably affect results. Although this outcome is disappointing, it is not too problematic as documents which have few training examples are likely to be small requirements documents making it easier to identify conflicts manually and without the aid of EA-Analyzer. However, when the documents are larger they are likely to have more training examples which will increase the accuracy of EA-Analyzer, importantly it is in these larger documents where identifying conflicts is more difficult and so using EA-Analyzer is more desirable.

On the other hand, although we achieve an accuracy of around 93%, there will still be about 7% of misclassified conflicts and harmonies. However, it is not the intention of this tool to fully replace a requirements analyst, but instead to assist him/her in identifying conflicts. Thus, the final decision on whether to accept or reject the suggested conflicts and harmonies for each set of requirements ultimately rests with the analyst. In fact, this is inevitable, as what is perceived as conflict in one system can be at harmony in another if the analyst deems it tolerable in the given context. Thus, the analyst should review the suggested classifications both for mistakes (misclassifications) and for the acceptability of the suggestions for the given system context.

5 Related Work

There is a large set of topics related to the work presented in this paper. We broadly group these into three main categories: work related to the use of NLP, work related to conflict identification in requirements, and work related to text classification with machine learning approaches.

There are several NLP-based approaches that use natural language as initial input for concept identification [42] [4] [13] [46] [18] and/or for deriving various models [1] [8] [4] [13]. For instance, EA-Miner [42] supports identification of crosscutting concerns in natural language requirements by matching these requirements against a lexicon dedicated to known types of crosscutting concerns (e.g. security, persistence, etc.). Theme/Doc [4] [13] uses lexical analysis for identification of interdependent activities (i.e. verbs) in NL requirements, as well as structuring verbs into sets of synonyms for major theme construction. In [18], the AbstFinder tool is used for identification of abstractions (i.e. concepts such as booking a flight) in requirements elicitation documents. CCVerbFinder [46] uses NLP for action verb-based feature identification in the code and related comments rather than requirements, though it is interesting to note that it also uses the notion of verb and object. The Circe environment [1] helps extract abstractions from natural language (Italian) texts, and build such models of the system as ER, DFD, OO design, etc. In the Conceptual Linguistically based Object oriented Representation language (COLOR-X) approach [8] the analyst has to select sentences for tagging in an annotation tool, then manually transform them into structured sentences as defined by set guidelines. The structured sentences are then automatically transformed into a specification language and represented as UML class diagram-like and state machine-like models. However, none of these approaches uses NL text for conflict identification, which is the main focus of our work.

There also are a number of RE approaches that focus on the issue of conflict identification [17] [34] [48] [32] [12]. For instance, the Viewpoints for Inconsistency Management (VIM) approach uses viewpoints - partial specifications of system requirements - and suggests to define a set of viewpoint consistency rules [17] [34] [33] to ensure system consistency upon viewpoint composition. Such rules are exemplified in the xlinkit tool [32] and are based on modeling formalisms (e.g., UML diagrams, etc.) and their augmentations (e.g., naming conventions, etc.) represented in XMI. The Knowledge Acquisition in Automated Specification (KAOS) approach [48] views requirements analysis as two coordinated tasks: requirements acquisition (where requirements are structured into system models, e.g., goal graphs) and formal specification [15]. The formally specified representation of requirements can then be checked for various conflicts. The Non-Functional Requirements Framework (NFRF)[12] supports identification of conflict points via correlation catalogues which help to examine the cross-impact of the softgoals and decide between competing alternative solutions. Here the non-functional requirements must be modeled in a goal graph. Thus, none of these approaches is designed for working with unconstrained natural language text.

The Naive Bayes approach has been applied to several text classification tasks, including classifying usenet news articles [20], classifying e-mail into folders [14] [35], identifying interesting news articles [24], and filtering spam e-mails [41]. The text classification problem has also been addressed with many machine learning techniques, such as nearest-neighbor methods [50], support vector machines (SVM) [21], and boosting [44]. However, none of these techniques have been applied to the problem of identifying conflicts in natural language specifications.

Thus, despite the existence of numerous related approaches, the problem of conflict identification using unrestricted NL as input (without any model formalization) has not been addressed to date.

6 Conclusions

AORE provides an effective way to modularize and compose concerns in requirements documents. AORE externalizes concern interactions and interdependencies in explicitly dedicated composition specifications. These composed concerns along with their compositions are an excellent starting point for analyzing conflicting dependencies. However, detecting conflicts in large natural language AO requirements documents could be an error-prone and time-consuming task due to the large number and complexity of the inter-dependencies to be considered. As discussed earlier, the formalization-, modeling-, and stakeholder-based approaches, developed to date in the AORE community, are unable to provide low effort and high precision techniques for conflict identification in textual AO requirements.

In this paper, we have presented EA-Analyzer - a tool which demonstrates that it is indeed possible to automate the process of detecting conflicts within textual AO requirements compositions. The tool is implemented as a novel application of the Naive Bayes learning method, in which the problem of detecting conflicts is formulated as a text classification problem.

Moreover, we have evaluated the tool via an empirical evaluation with three industrial-strength requirements documents and a well established academic case-study used in the AO research community. This empirical evaluation has shown that conflicting dependencies can be detected with a high accuracy, provided that the training set has a sufficient number of examples. Our future work will focus on evaluating the EA-Analyzer tool with other requirements documents from different domains to validate the generalization power of the Naive Bayes classifier. In future evaluations we will also test a number of other classifiers, such as SVM [6] and nearest-neighbor methods [6], to identify the best machine learning approach for detecting conflicts. Also, we will investigate how other AO approaches, such as ARCADE [38] or CORE [29], can be supported by the tool.

Thus, with this first tool for automated conflict identification in textual AO requirements and compositions, we demonstrate that the power of AORE to represent concern inter-relationships knowledge can be effectively harvested for conflict detection and analysis. We see this work as the stepping stone towards effort reduction in AO requirements conflict identification, and supporting application of advanced modularity and analysis in textual requirements.

7 Acknowledgements

This work has been supported by a Marie Curie Fellowship from the European Commission (Grant Agreement No. PIIF-GA-2008-221016) and FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

References

[1] V. Ambriola, V. Gervasi, On the systematic analysis of natural language requirements with circe, Automated Software Engg. 13 (1) (2006) 107–167.

[2] J. Araújo, J. Whittle, D.-K. Kim, Modeling and composing scenario-based requirements with aspects, in: RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International, IEEE Computer Society, Washington, DC, USA, 2004.

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

[4] E. Baniassad, S. Clarke, 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, DC, USA, 2004.

[5] O. Barais, J. Klein, B. Baudry, A. Jackson, S. Clarke, 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, DC, USA, 2008.

[6] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[7] I. S. Brito, F. Vieira, A. Moreira, R. Ribeiro, Handling conflicts in aspectual requirements compositions, Transactions on Aspect Oriented Software Development (TAOSD).

[8] J. F. M. Burg, R. P. v. d. Riet, Color-x: Linguistically-based event modeling: A general approach to dynamic modeling, in: CAiSe '95: Proceedings of the 7th International Conference on Advanced Information Systems Engineering, Springer-Verlag, London, UK, 1995.

[9] R. Chitchyan, Semantics-based composition for aspect-oriented requirements engineering, Ph.D. thesis, Computing Department, Lancaster University (2007).

[10] R. Chitchyan, P. Greenwood, A. Sampaio, A. Rashid, A. Garcia, L. Fernandes da Silva, Semantic vs. syntactic compositions in aspect-oriented requirements engineering: an empirical study, in: AOSD '09: Proceedings of the 8th ACM international conference on Aspect-oriented software development, ACM, New York, NY, USA, 2009.

[11] 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.

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

[13] S. Clarke, E. Baniassad, Aspect-Oriented Analysis and Design, Addison-Wesley Professional, 2005.

[14] W. W. Cohen, Learning rules that classify e-mail, in: In Papers from the AAAI Spring Symposium on Machine Learning in Information Access, AAAI Press, 1996.

[15] A. Dardenne, A. van Lamsweerde, S. Fickas, Goal-directed requirements acquisition, Sci. Comput. Program. 20 (1-2) (1993) 3–50.

[16] P. Domingos, M. J. Pazzani, On the optimality of the simple bayesian classifier under zero-one loss, Machine Learning 29 (2-3) (1997) 103–130.

[17] S. Easterbrook, B. Nuseibeh, Using viewpoints for inconsistency management, Software Engineering Journal 11 (1996) 31–43.

[18] L. Goldin, D. M. Berry, AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation, Automated Software Engg. 4 (4) (1997) 375–412.

[19] J. Gray, D. P. Siewiorek, High-availability computer systems, Computer 24 (9) (1991) 39–48.

[20] T. Joachims, A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, in: ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[21] T. Joachims, Text categorization with suport vector machines: Learning with many relevant features, in: ECML '98: Proceedings of the 10th European Conference on Machine Learning, Springer-Verlag, London, UK, 1998.

[22] J. Kienzle, N. Guelfi, S. Mustafiz, Crisis management systems: A case study for aspect-oriented modeling, in: S. Katz, M. Mezini, J. Kienzle (eds.), Transactions on Aspect-Oriented Software Development VII, vol. 6210 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 1–22.

[23] 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.

[24] K. Lang, Newsweeder: Learning to filter netnews, in: in Proceedings of the 12th International Machine Learning Conference (ML95, 1995.

[25] D. Lewis, Naive (bayes) at forty: The independence assumption in information retrieval, in: C. Ndellec, C. Rouveirol (eds.), Machine Learning: ECML-98, vol. 1398 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1998, pp. 4–15, 10.1007/BFb0026666.
URL `http://dx.doi.org/10.1007/BFb0026666`

[26] A. McCallum, K. Nigam, A comparison of event models for naive bayes text classification, in: AAAI-98 workshop on learning for text categorization, 1998.

[27] K. Mehner, M. Monga, G. Taentzer, Interaction analysis in aspect-oriented models, in: RE '06: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), IEEE Computer Society, Washington, DC, USA, 2006.

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

[29] A. Moreira, J. Araújo, A. Rashid, A concern-oriented requirements engineering model, in: Proceedings of CAiSE 2005, LNCS, Vol. 3520, 2005.

[30] 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.

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

[32] C. Nentwich, W. Emmerich, A. Finkelstein, E. Ellmer, Flexible consistency checking, ACM Trans. Softw. Eng. Methodol. 12 (1) (2003) 28–63.

[33] B. Nuseibeh, S. Easterbrook, A. Russo, Making inconsistency respectable in software development, Journal of Systems and Software 58 (2001) 171–180.

[34] B. Nuseibeh, J. Kramer, A. Finkelstein, Viewpoints: meaningful relationships are difficult!, in: ICSE '03: Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, 2003.

[35] T. R. Payne, P. Edwards, Interface agents that learn: An investigation of learning issues in a mail agent interface (1995).

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

[37] 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.

[38] 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.

[39] P. Rayson, Wmatrix.
URL `http://www.comp.lancs.ac.uk/ucrel/wmatrix/`

[40] Y. R. Reddy, S. Ghosh, R. B. France, G. Straw, J. M. Bieman, N. McEachen, E. Song, G. Georg, Directives for composing aspect-

oriented design class models., Trans. Aspect-Oriented Software Development (2006) 75–105.

[41] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A bayesian approach to filtering junk e-mail, in: AAAI'98 Workshop on Learning for Text Categorization, 1998.

[42] A. Sampaio, R. Chitchyan, A. Rashid, P. Rayson, 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, NY, USA, 2005.

[43] 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.

[44] R. E. Schapire, Y. Singer, Boostexter: A boosting-based systemfor text categorization, Mach. Learn. 39 (2-3) (2000) 135–168.

[45] P. Shaker, D. K. Peters, Design-level detection of interactions in aspect-oriented systems, in: Proceedings of the Aspects, Dependencies, and Interactions Workshop at ECOOP 2006, 2006.

[46] D. Shepherd, L. Pollock, K. Vijay-Shanker, Towards supporting on-demand virtual remodularization using program graphs, in: AOSD '06: Proceedings of the 5th international conference on Aspect-oriented software development, ACM, New York, NY, USA, 2006.

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

[48] A. van Lamsweerde, A. Dardenne, B. Delcourt, F. Dubisy, The KAOS project: Knowledge acquisition in automated specification of software, in: Proceedings AAAI Spring Symposium Series, Stanford University, American Association for Artificial Intelligence, 1991.

[49] 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.

[50] Y. Yang, C. G. Chute, An example-based mapping method for text categorization and retrieval, ACM Trans. Inf. Syst. 12 (3) (1994) 252–277.

A Naive Bayes - The Bayesian Approach in EA-Analyzer

The problem of detecting conflicts in EA-Analyzer is formulated as a classification problem. Thus, 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 $V = \{Conflict, Harmony\}$. The aim of the Bayesian approach [28] is to assign the most probable target value, v_{NB}, given the input vector of features $< a_1, a_2, ..., a_n >$:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j|a_1, a_2, ..., a_n) \tag{5}$$

Equation 6 presents the Bayes theorem:

$$P(v_j|a_1, a_2, ..., a_n) = \frac{P(a_1, a_2, ..., a_n|v_j)P(v_j)}{P(a_1, a_2, ..., a_n)} \tag{6}$$

Hence, from the Bayes theorem in Equation 6, we can rewrite Equation 5 as follows:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2, ..., a_n|v_j)P(v_j)}{P(a_1, a_2, ..., a_n)} \tag{7}$$

Since the denominator in Equation 7 does not depend on v_j and the values of $a_1, a_2, ..., a_n$ are given, this denominator is a constant independent of v_j. Thus, we can rewrite Equation 7 as follows:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2, ..., a_n|v_j)P(v_j) \tag{8}$$

The Naive Bayes approach assumes that the features are conditionally independent given v_j. In other words, the probability of observing $a_1, a_2, ..., a_n$ is the product of the probabilities of each individual feature:

$$P(a_1, a_2, ..., a_n|v_j) = \prod_i P(a_i|v_j) \tag{9}$$

Hence, we can rewrite Equation 8 as follows:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j) \tag{10}$$