Mining Aspects in Requirements

Américo Sampaio, Neil Loughran, Awais Rashid and Paul Rayson
Computing Department, Lancaster University, Lancaster, UK
{a.sampaio, loughran, marash, paul}@comp.lancs.ac.uk

Abstract

The early identification and documentation of crosscutting concerns enables better change management and traceability of requirements. Moreover, this also improves the early identification of candidate aspects in the design and implementation stages. Current techniques for identifying aspects in requirements are ineffective when requirements are complex or unstructured. This paper describes an approach that utilises corpus-based natural language processing (NLP) techniques to effectively enable the identification of aspects in a semi-automated way. The technique proposed here describes how unstructured sources of requirements (e.g., interviews, natural language description of the system) or requirements documents can be automatically mined to help the requirements engineer quickly identify and build a structured aspect-oriented model of the requirements.

1. Introduction

A challenging dilemma faced by the software engineering community is how to deliver quality software when severe time constraints are imposed [6]. In many cases, developers resort to ad-hoc "short cuts" that accelerate the development process, but suffer from lack of structure, impacting the quality of the process and its deliverables.

Aspect-oriented software development (AOSD) is a technique that has shown encouraging results in improving modularization of software systems, therefore enhancing evolution and lowering time to market. In order to maximize its benefits, AOSD should be used from the early stages of software development such as domain analysis and requirements engineering [1- 4].

An important step towards effectively supporting early aspects identification is to provide not only abstractions that represent crosscutting requirements, but also to offer mechanisms (e.g., tool support) for mining aspects in requirements documents [1, 4]. This helps the requirements engineer to identify aspectual requirements and their relationships with other requirements.

However, current techniques for mining aspects in requirements do not provide an effective approach when requirements are complex or unstructured. For instance, the Theme/Doc [1] approach provides a tool for semi-automatic identification of crosscutting behaviours in requirements specifications. The identification process is based on a lexical analysis of the requirements document, which searches the document for some keywords provided by the developer. The tool automatically produces a graphical view that maps the relationships between the behaviours, which can help the developer to identify the aspect candidates.

Despite presenting interesting ideas on how to identify and model crosscutting concerns, this approach relies on an inadequate and sometimes inefficient way of analysing the document. The first problem is that the analyst has to read all the requirements document to input the keywords which can be time consuming and unrealistic for complex and time-constrained projects. Moreover, the tool presupposes that the requirements document is described in a certain way suited to fit the lexical analyser. Therefore, these limitations impose serious issues when the problem is complex and development is document centric, where requirements can be obtained from various sources such as documented interviews with stakeholders, legacy documents and extensive requirements documents (e.g., thousands of textual pages).

The other approach presented in [4] focuses on using information retrieval techniques for mining specific aspects in requirements documents. In this approach, the analyst uses the tool to search places in the document where some crosscutting influence occurs. The search is based on the analyst's assumption of some concern that s/he thinks should be crosscutting (e.g., performance), and then, highlights only specific parts of the requirements document that match the criteria.

Therefore, this approach, when compared to Theme/Doc, saves time for finding requirements crosscut by some specific concern. However, Theme/Doc is more suitable with the task of finding all the crosscutting influences and modelling their relationships.

Our paper describes an approach that builds upon the ideas presented in the two previous approaches, but utilises corpus-based [11] natural language processing (NLP) techniques in order to effectively enable the identification of aspects in a semi-automated way. The proposed technique describes how unstructured sources of requirements (e.g., interviews, natural

language descriptions of the system) or requirements documents can be automatically mined, thus allowing the requirements engineer to quickly develop a structured aspect-oriented model of the requirements.

The main goal of our approach is to determine potential aspect candidates in requirements documents regardless of how they are structured. The approach uses NLP techniques which provide support for context sensitive analysis of requirements [5]. Tool support is provided to help the developer automatically mine and model the crosscutting concerns without having to previously read the requirements documents.

The remainder of this paper is described as follows. Section 2 describes the approach proposed and how NLP is used. Section 3 shows a small example and outlines how tool support can help developers mine aspects. Finally, Section 4 concludes the paper.

2. Approach for Mining Aspects

Requirements can be obtained from many different sources such as interviews with stakeholders, standards, ordinary documents and legacy systems. Moreover, requirements documents can be expressed in many ways ranging from natural language descriptions to more structured methods such as use cases [8], viewpoints [9] and formal specifications [10]. Each approach has its own merits and demerits and can be better suited for a specific purpose or situation.

Figure 1 shows our approach of identifying and representing aspects in requirements documents. The main advantages of our approach are:

- It can work with any kind of textual documents regardless of their structure (interviews, natural language descriptions, use case textual descriptions, etc.).
- The most time consuming activities, such as identifying concerns, viewpoints and action words, are partially automated.

Nearly all applications of NLP to requirements engineering have used rule-based techniques. This severely restricts their applicability to well-formed requirements documents that use a controlled subset of natural language and means that (for example) uncontrolled text such as transcripts of stakeholder interviews are intractable to this kind of processing. In practice, fully automated synthesis of requirements from such documents is infeasible. However, Goldin and Berry [7] and our own work on the REVERE project [5] have successfully used NLP to help the analyst identify important concepts (objects, agents, functions etc). REVERE's tools are based on underlying statistical techniques which enable them to be robust on uncontrolled text. WMATRIX [5] uses a combination of part-of-speech and semantic tagging, frequency analysis and concordances to identify domain concepts of potential significance. Part-of-speech analysis automates the extraction of nouns and verbs from the text. Semantic analysis groups related words and multi-word expressions into concepts even when many different word forms are used in the documents, e.g., *vehicle, vehicles*, *driver*, *drivers* and *traffic* grouped into the semantic field 'land transport' from the example below (Section 3).

The approach begins (*Phase 1*) by analysing existing documents that are sources for requirements elicitation such as interviews done with stakeholders (e.g., clients, managers, users) or informal descriptions of the system. The mining tool reads these files and passes them to WMATRIX, which can produce analyses that helps to identify concerns and viewpoints using natural language processing techniques. The mining tool enables tailoring the information that flows in and out from WMATRIX, for example, showing only a subset of the action words (verbs identified by WMATRIX) based on some criteria.

At the end of *Phase 1* a more structured requirements specification (Intermediate Model) can be produced as an output. It is important to note that this specification is not fully automated by the tools, whose main focus is to help the requirements engineer in discovering concerns and viewpoints in order to have a more structured description of the requirements.

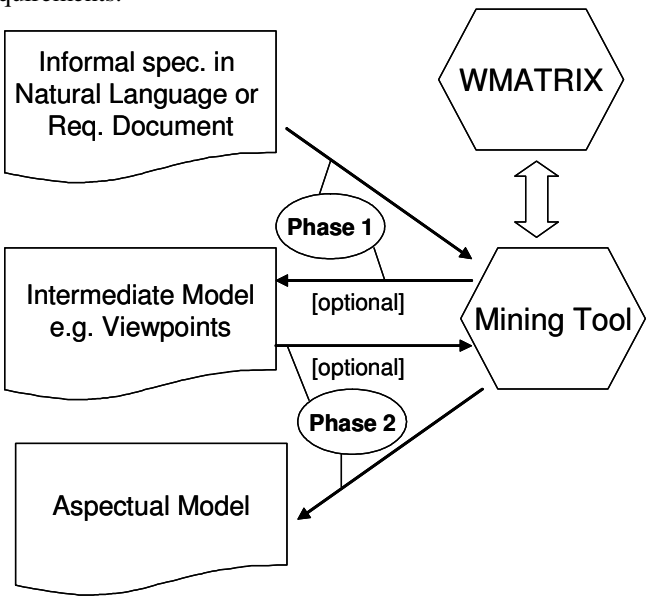

Figure 1 - Model for Aspect Mining

The next phase (*Phase 2*) is to read directly from the informal files, or from the intermediate model, and process the information in order to search for crosscutting requirements and candidate aspects. The mining tool has the role of setting the criteria (e.g., applying filters to send/receive information to/from WMATRIX) that will be used by WMATRIX to process the information. Moreover, the mining tool can filter the results in order to identify what the candidate aspects are and produce a model that represents the relationships between the requirements. In phase 2 we can reuse the intermediate viewpoint model. The mining tool (via WMATRIX) examines the dispersion of candidate aspects across the various requirements. If a candidate aspect is well dispersed - i.e.

appears in many requirements, then the case for that candidate aspect is stronger.

Our approach for mining aspects can be used regardless of the structure of the textual document provided as input (e.g., informal descriptions, interviews, structured documents). Document-heavy domains based on regulations, standards and various types of extensive documentation can thus benefit from our approach. The tools enable accelerated development because they do not impose any kind of specific format and do not depend on previous knowledge on the requirements by the requirements engineer.

The tools enable the developer to quickly mine the requirements and gain an overall understanding of them, for example, using semi-automatic features for producing an intermediate model using viewpoints. Moreover, the tools also help to semi-automatically mine for crosscutting requirements, enabling the early identification and separation of concerns, and, therefore, benefiting following stages. It is important to mention that the tools require the intervention of the requirements engineer for creating the models. The next section shows how the tools can be used and how their capabilities can enhance development.

3. Example

In this section we give a small example to show how the approach described in Section 2 works. The system described is a simplified version of the toll collection system on the Portuguese highways [2, 3]. The system has the following informal description:

"In a road traffic pricing system, drivers of authorised vehicles are charged at toll gates automatically. The gates are placed at special lanes called green lanes. A driver has to install a device (a gizmo) in his/her vehicle. The registration of authorised vehicles includes the owner's personal data, bank account number and vehicle details. The gizmo is sent to the client to be activated using an ATM that informs the system upon gizmo activation. A gizmo is read by the toll gate sensors. The information read is stored by the system and used to debit the respective account. When an authorised vehicle passes through a green lane, a green light is turned on, and the amount being debited is displayed. If an unauthorised vehicle passes through it, a yellow light is turned on and a camera takes a photo of the plate (used to fine the owner of the vehicle). There are three types of toll gates: single toll, where the same type of vehicles pay a fixed amount, entry toll to enter a motorway and exit toll to leave it. The amount paid on motorways depends on the type of the vehicle and the distance traveled".

A file containing the textual description above is then provided as input to the mining tool that communicates with the natural language processor (WMATRIX). WMATRIX provides support in the following tasks:

I. Identifying Viewpoints: WMATRIX has been used before in [5] to identify stakeholders in requirements documents. The criteria used there, was to identify human agent nouns with common endings for job titles (such as 'er' or 'et' or 'or' or 'man') such as controller, pilot, etc. This can be extended to identify other kinds of viewpoints that are not human such as gizmo, exit toll, ATM (In the toll system). The list of candidate viewpoints is produced automatically and shown to the requirements engineer so that s/he can select the relevant ones and produce the intermediate model described in Section 2. Moreover, the mining tool can also insert tags in the text that help the engineer to quickly find the requirements related to the viewpoints.

Figure 2 shows how WMATRIX is used to analyse the informal description given above and list the candidate viewpoints.

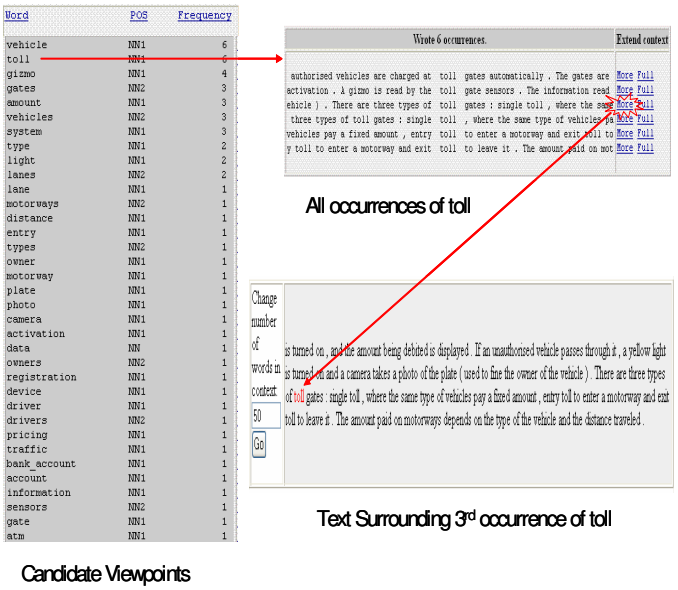

Figure 2 - Viewpoints Identification

Candidate Viewpoints are automatically identified by WMATRIX by listing the nouns in the text alongside their part of speech (POS) class (e.g., NN1 – Singular nouns, NN2 – Plural nouns) and frequency of occurrence. The user is able to view parts of small sentences where all occurrences of a specific word occur, such as "toll". Moreover, s/he has the option of viewing a more detailed slice of text for each occurrence of the word "toll".

These features enable the user to quickly mine the viewpoints and their related requirements as s/he has to focus on reading only small parts of the text as shown next for the toll viewpoint.

Viewpoint: Toll.
Requirements:
1. Vehicles are charged at toll gates automatically.
2. A gizmo is read by the toll gate sensors.
3. There are three types of toll gates: single toll, where the same type of vehicles pay a fixed amount, entry toll to enter a motorway and exit toll to leave it.

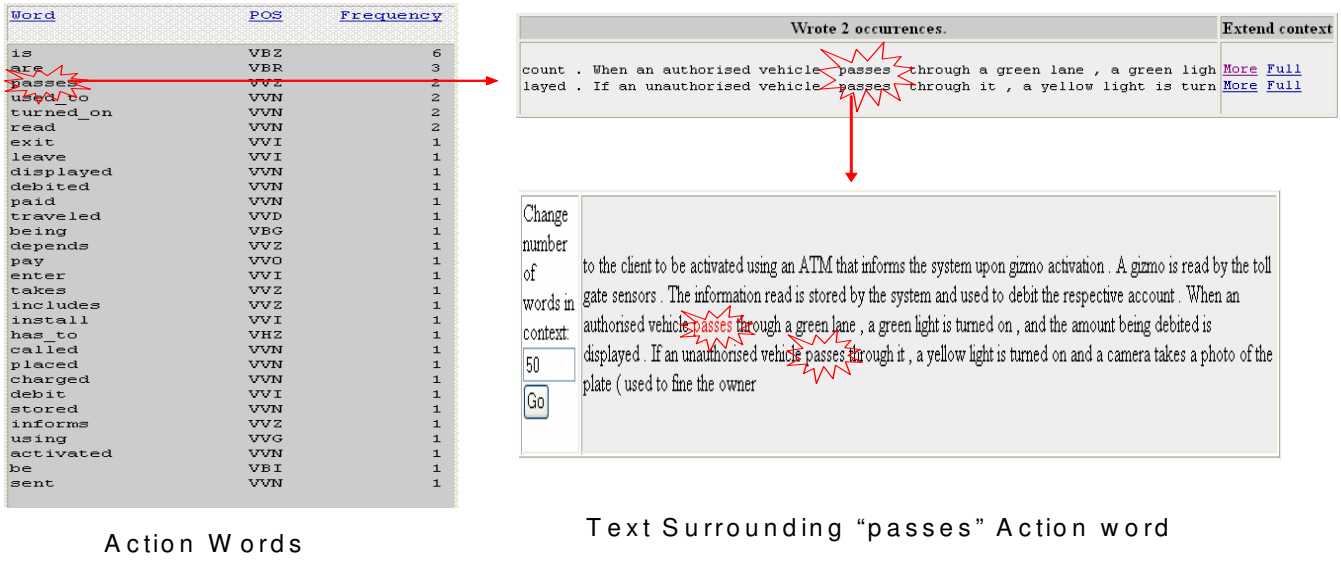

Figure 3 - Action Words Identification

Moreover, some candidate viewpoints can be immediately discarded by the user, as they do not comply well with the concept of a viewpoint (e.g., type and data). Therefore, the tool enables the user to quickly scan through the text reading only relevant information, providing him/her not only requirements' understanding, but also an effective way of producing a more structured requirements description using viewpoints.

II. Identifying Crosscutting Requirements: This task can be supported in different ways by the tools. In one approach, similar to that of Theme/Doc, the tool can automatically look for action words, which can be identified as verbs by the WMATRIX tool, and produce a model that represents the relationships between the requirements. The engineer does not have to provide the action words in advance and, also, the natural language processor enables a more context sensitive analysis of the words recognising for example, that the actions “collect” and “pick up” are in the same semantic field.

Figure 3 shows how WMATRIX is used to list the action words recognized as verbs in the text of our example requirements specification.

The tool provides features for filtering the previous list by, for example, excluding verbs that do not represent an action such as auxiliary verbs (is, are, be). The list can also be filtered to present, as a single action, different verbs that have the same semantic meaning in the context. Action words (e.g., passes) and their relating requirements (surrounding phrases such as “authorised vehicle passes through a green lane”) can be automatically identified by the tools in order to produce a model that represents their relationships. Such a model can thus help the developer to identify and model crosscutting requirements.

Another approach to identify crosscutting influences could be to set filters which search for known classes of words that map to non-functional requirements such as security, persistence and performance and suggest them as concerns as shown in [2, 3]. The problem with this approach is that sometimes these concerns are implicit and difficult to automatically mine, relying more on the judgment of the developers.

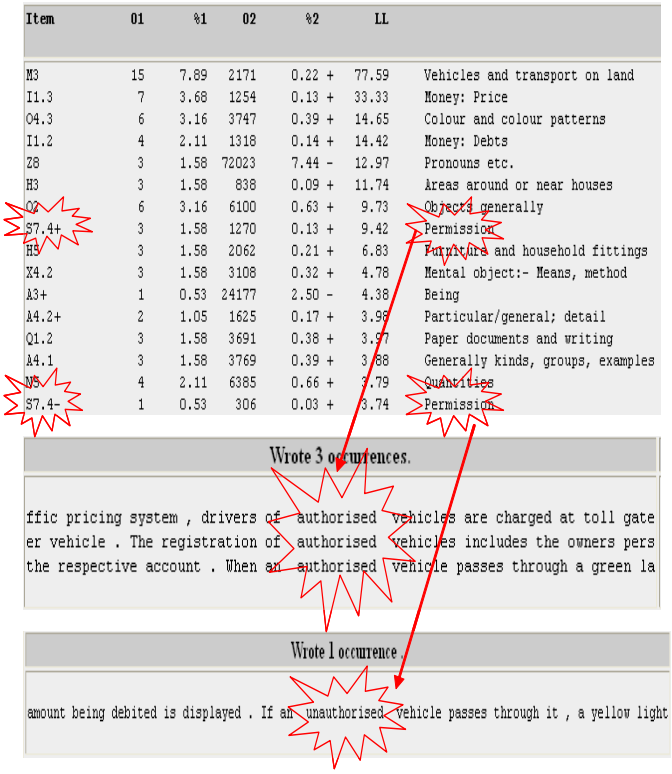

Figure 4 - Concern Identification by Semantic Analysis

Figure 4 shows an example of how semantic analysis can be performed to group words by their semantic classes. For example, the tool identified 3 occurrences of the semantic

class S7.4, which means *permission*, with a positive meaning. The three occurrences refer to the word "authorised" and are shown in Figure 4. This can help the developer identify some concerns related to the authorization of vehicles such as security and correctness. The system must ensure that authorized vehicles are detected and proper actions are taken. The bottom of the figure shows the identification of the same semantic class but with a negative meaning. This suggests that a different set of actions has to be considered for "unauthorised" vehicles.

The examples described in this section point out the capabilities provided by our approach of mining aspects in requirements. It is important to mention that the mining tool is still in an early stage of development, but the capabilities provided by the WMATRIX tool shown in [5] make us believe that the approach described previously is feasible due to the effectiveness of NLP in context-sensitive analysis.

4. Conclusions

This paper has proposed an approach for mining aspects from requirements-related documents. The approach is based on NLP techniques that enable an efficient context sensitive analysis of textual documents. Documents analysed by the tools can vary from very informal textual documents, such as interviews and high level descriptions of the system, to more structured documents such as use case textual descriptions or viewpoint descriptions.

The approach suggests an optional step for producing a more structured description of the system, using viewpoints, which can be partially automated. However, the main goal of the tool is to provide partially automated support for aspect mining through concern identification or action word mapping.

Our future work will focus on completing the mining tool and applying it in a practical project. The tool will focus on providing support for automating the approach presented in this paper by using the features provided in the WMATRIX tool.

Acknowledgement: This work is supported by European Commission grant IST-2-004349: European Network of Excellence on Aspect-Oriented Software Development (AOSD-Europe), 2004-2008.

References

[1] E. Baniassad and S. Clarke. Finding Aspects in Requirements with Theme/Doc. In Proceedings of Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design, Lancaster, UK, 22 Mar. 2004.

[2] A. Rashid, A. Moreira, and J. Araujo. Modularisation and Composition of Aspectual Requirements. In Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, pages 11–20. ACM Press, 2003.

[3] A. Rashid, P. Sawyer, A. Moreira, and J. Araujo. Early Aspects: A Model for Aspect-Oriented Requirements Engineering. In Proceedings of IEEE Joint International Conference on Requirements Engineering (RE 2002), pages 199–202. IEEE Computer Society, 2002.

[4] L. Rosenhainer. Identifying Crosscutting Concerns in Requirements Specifications. In Proceedings of OOPSLA Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop. October 2004, Vancouver, Canada.

[5] P. Sawyer, P. Rayson, and R. Garside. REVERE: support for requirements synthesis from documents. Information Systems Frontiers Journal. Volume 4, issue 3, Kluwer, Netherlands, pp. 343 – 353, 2000.

[6] Agile Alliance Website. Availabe at: http://www.agilealliance.org/home.

[7] L. Goldin, D. Berry, AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation, Automated Software Engineering, 4, 1997.

[8] I. Jacobson. Object-Oriented Software Engineering: A Use Case Driven Approach. 1st Edition, Addison-Wesley, 1992.

[9] I. Sommerville and P. Sawyer. Requirements Engineering – A Good Practice Guide. Wiley, 1997.

[10] B. Potter et al. An Introduction to Formal Specification and Z. Prentice Hall, 1991.

[11] R. Garside et al. Corpus Annotation: Linguistic Information from Computer Text Corpora. Addison Wesley Longman,1997.