International Journal of the Physical Sciences Vol. 6(11), pp. 2718-2730, 4 June, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.015
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

An approach for crosscutting concern identification at requirements level using NLP

Busyairah Syd Ali* and Zarinah M. D. Kasirun

[1]Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia.

Accepted 04 April, 2011

Poor requirements analysis process results in incomplete software applications. Some requirements appear as scattered and tangled concerns within requirements document. Hence it is difficult to identify such requirements. A number of research approaches such as Theme/Doc, early aspects identification, information retrieval and aspects identification using UML have been developed to identify crosscutting concern at the requirements level. Nevertheless, these approaches are only supported by semi-automated tools whereby human intervention is required to achieve the desired results. This research focuses on developing a tool to automatically identify crosscutting concern at the requirements level. A model based on Theme/Doc and early aspects identification approaches is formulated as the basis of this tool, 3CI. 3CI adopts natural language processing (NLP) techniques such as verb frequency analysis, part-of-speech tagging and dominant verb analysis. The tool usability, efficiency and scalability are evaluated by comparing the performance of a requirements engineer conducting similar task manually. Our evaluation on 3CI demonstrates 75% of accuracy.

Key words: Aspects-oriented requirements engineering, 3CI, crosscutting concern, dominant verb analysis.

INTRODUCTION

Aspects oriented requirements engineering (AORE) is relatively a new area of research under the requirements engineering (RE) domain. AORE aims at addressing crosscutting concerns by providing means for identification, modularization, composition as well as analysis of their influence on other requirements in the specification documents. According to Brito (2004), there are certain requirements that cannot be identified and modularized by the existing technique such as object oriented analysis. These requirements can present functional or non-functional requirement. She also claimed that these requirements crosscut or influence other requirements. They are scattered among other requirements and tangled within a requirement. Their characteristic varies. Some are obvious and some are subtle. Therefore it is difficult to identify them. These requirements are called “crosscutting concern”. Identification of crosscutting concern is a tedious task. Mining crosscutting concerns involves large volume of specification documents. Documents such as interview transcripts are usually inaccurate, full of perceptible contradictions and missing vital information. Furthermore crosscutting concerns are often scattered across a document making their identification difficult. Making it worse, sometimes similar requirements occur in different parts of the document paraphrased in different words. This issue is an unresolved problem for software developers. During the requirements analysis phase, it is difficult to see how the requirements are influenced by each other and how it will impact the whole software development process. If such concerns are not identified and modularized early enough it will affect the choice of software architecture. Then, it will be too late to reverse to the earlier process. This will increase the cost, time and effort. Therefore, it is important to identify crosscutting concern at the early stage so that it is not overlooked in the subsequent phases.

*Corresponding author. E-mail: bush@um.edu.my. Tel: (+44) 7400915527. Fax: (+44) 2075946102.

The emphasis of this research is to formulate a model to automatically identify crosscutting concern in English-based specifications document and materialize the model by developing a tool called 3CI. Finally, testing and evaluation is conducted to verify the tool usability, efficiency, scalability and accuracy. Other details such as: the definition of "crosscutting concern" and 'aspects", the description of the proposed model for an automated approach to identify crosscutting concerns, the tool designed based on the proposed model, the tool, the existing approaches with the proposed approach to identify the crosscutting concern were discussed in this paper.

CROSSCUTTING CONCERN AND ASPECTS

Sutton and Rouvellou (2002) defined *concern* as "any matters of interest in a software system". It can be directly related to the system or its environment. We can also define a requirement as a concern stated by the system users or stakeholders. The added crosscutting concern can either be functional or non-functional requirements.

When a concern crosscuts one or more of other concerns, they are called crosscutting concern. For example, in case of two requirements 'A' and 'B', an act of software enhancement is initiated in which 'B' cannot be satisfied without affecting 'A' that means requirement 'A' crosscuts requirement 'B'. In this case the requirement that crosscuts others are referred to as being crosscutting concern, which is 'A'. A number of researchers also address crosscutting concern as *candidate aspect* (Araujo et al., 2002). Based on the research and literature review, we conclude that crosscutting concerns have tangled, scattered, intertwined and interdependent behaviours among and within the requirements specifications. This is illustrated in Figure 1. Some crosscutting concerns can be obvious and can be easily identified. Often crosscutting concerns are subtle, so it is difficult to identify them. Once they are identified, they are encapsulated into modules called *aspect*. According to Rosenhainer (2004), crosscutting *influence* indicates the relationship between two or more requirements which is established by one crosscutting the other. For example if 'A' has crosscutting influence on 'B' it means 'A' crosscuts 'B'. Crosscutting influence denotes dependency among requirements.

THE MODEL

Here we describe a model that automatically identifies crosscutting concern as well as their crosscutting relationships at the requirement level. The model is depicted in Figure 2. It is formulated based on Theme/Doc Approach (Baniassad and Clark, 2004) and EA-Miner (Sampaio et al., 2005). It utilizes NLP techniques to extract linguistic properties in each unique requirement and exploits these properties to identify crosscutting concerns relationships in a requirements document. Natural language processing (NLP) is identified as an effective solution to cater for this problem. NLP techniques such as part-of-speech analysis, word frequency analysis and dominant verb analysis contribute in the processing of requirements phrases to assist aspects mining. Ali and Kasirun (2008b) described further on the model. In order to realize the model a tool, 3CI, is developed. Detailed description on the 3CI tool.

Model description

The processes in the model are depicted in Figure 2.

Structure requirements

This task involves numbering all the requirements agreed by the stakeholders. This is required to identify and manipulate each requirement uniquely in the next stages.

Part of speech analysis (POS)

Part of speech tagger tags each word in the requirements phrase by categorizing the words in a text to a particular part of speech based on both its definition, as well as its context. The part of speech categories can be nouns, verbs, adjectives, adverbs and etc.

Verb frequency analysis

The frequency of occurrence of each verb identified by POS will be calculated to show its dispersion (scattering) throughout the document. Higher level of dispersion indicates the strength of the verb as candidate aspect. This information identifies the verbs triggered in more than one situation as described in the requirements. Corresponding verbs will be used for modeling the relation with the requirements and interdependency among other verbs.

Semantic analysis

This task utilizes semantic tagger to analyse and identify the verbs used in a similar context representing the same requirement concern.

Filter verbs identified

Based on the semantic analysis performed, duplication of the verbs in terms of the context is discarded. For example the usages of 'protect' and 'secure' in the same context allow us to discard one of the terms since it refers to the same meaning.

Map relationship view

To show the crosscutting relationship, we map the requirements using a matrix as shown in Table 1 to identify the requirements {R1..Rn} influenced by corresponding verbs {v1..vn}. For example R4 is influenced by v1, v2, v3 and v4. Therefore there are

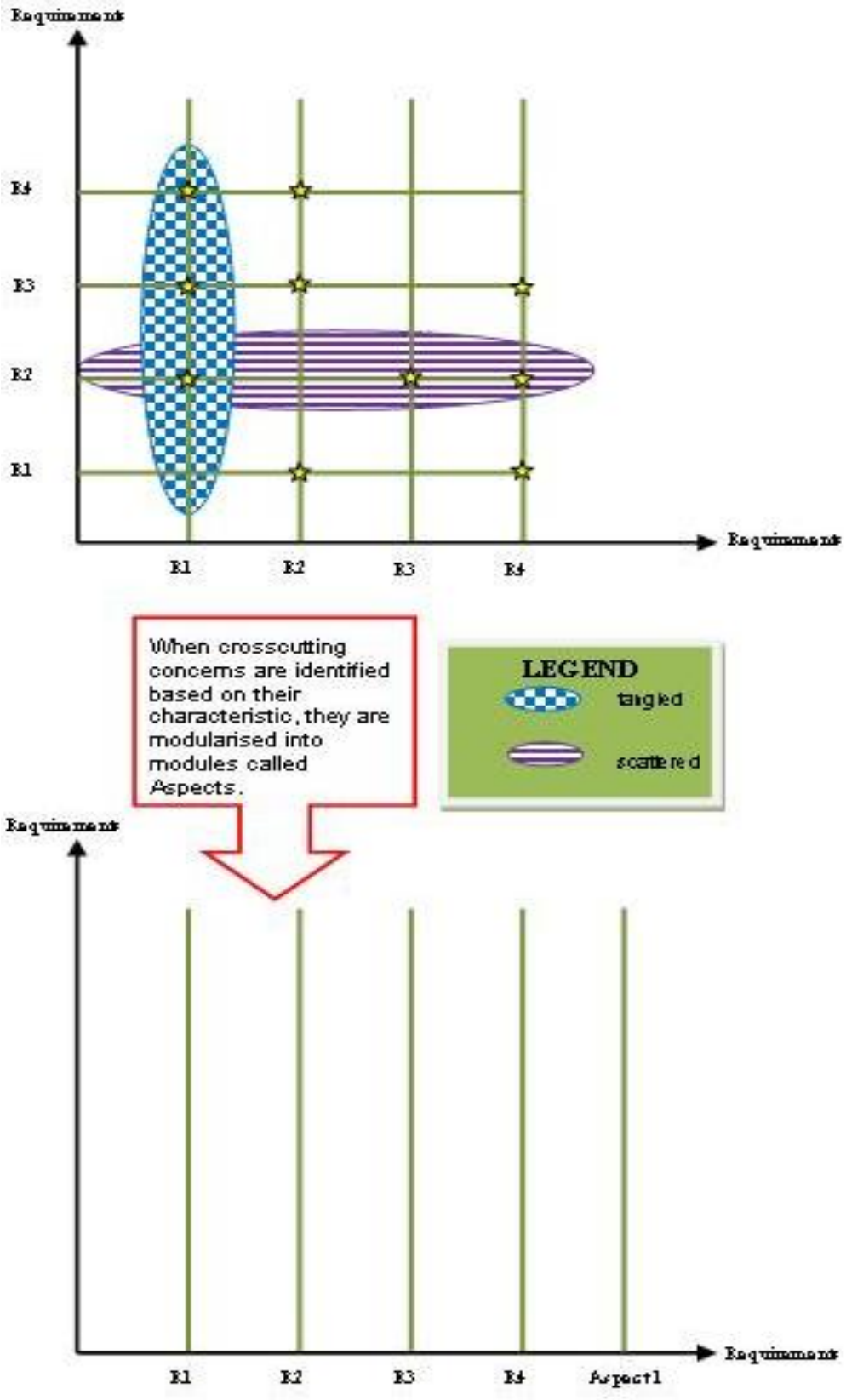

Figure 1. An Illustration of crosscutting concern behaviours in requirements specification document.

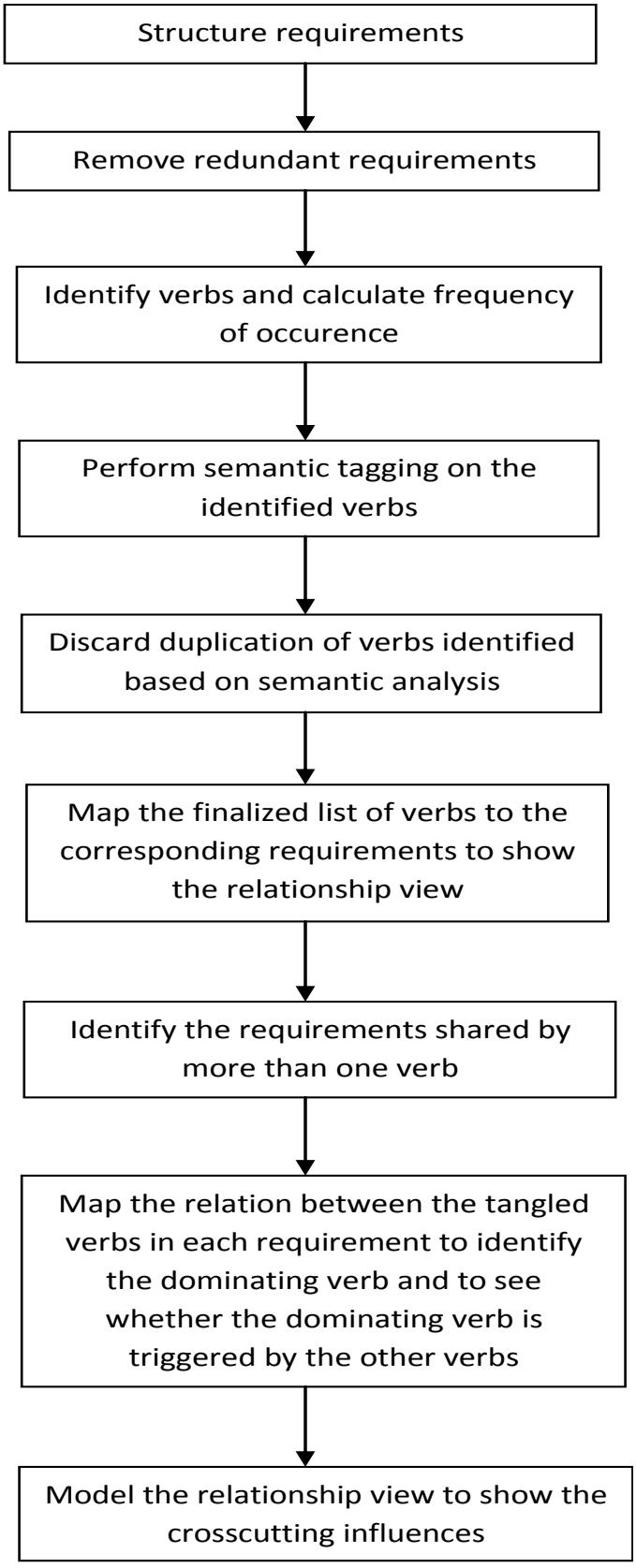

Figure 2. Crosscutting concern identification model.

descriptions of four verbs tangled within requirement R4. The matrix can become quite large if there are many requirements and many verbs. This can be mitigated by imposing constraints in the next stage.

Refining the relationship view

Based on the relationship view, the requirements shared by more than one verb are identified. As shown in Table 1, R4 is an example. The matrix is then refined by showing the requirement shared by more than one verb and all the requirements influenced by the verbs identified in the shared requirement. The refined matrix is tabulated in Table 2.

Dominant verb analysis

A matrix as shown in Table 3 is regenerated to map the relationship between the tangled verbs in each requirement to identify the dominating verb in the requirement and to see if the dominating verb is triggered by the other verbs. The dominating verb is the candidate aspect. Dominant verb analysis is performed on each requirement identified in the refined relationship view. Table 3 shows that in the specified requirement, there are four verbs identified v1, v2, v3 and v4. The verb 'v1' is triggered by verb 'v2', 'v3' and 'v4'.

Syntactic rules to identify dominating verb

Types of syntax for shared requirements clauses

By analyzing numerous requirements clauses that is shared by more than one verb, a few type of morphology are identified. The clauses shown in the matrix tabulated in Table 2, have these types of morphologies. However, it may not be limited to these syntactic categories only. At this point of research, the syntax is limited to these only. The conjunction words considered are *CONJ [if, when, that, having].*

Syntax 1 → NP—VP1—"to"—VP2—CONJ—VP3:

Description*:* NP means noun phrase, the long ("—")indicates precedence, whereby the noun phrase precedes the VP (verb phrase) which precedes the word "to" that precedes a verb phrase followed by any CONJ(conjunction) as stated above which then precedes another verb phrase. In this morphology, it shows that VP2 is triggered by VP1 and VP3.

Example*:* The students are allowed to enter if they have registered.

Syntax 2 → NP—VP1—CONJ—VP2:

Description*:* In this syntax, noun phrase precedes the verb phrase which precedes a conjunction that's finally precedes another verb phrase. The morphology in this syntax indicates that VP1 is triggered by VP2.

Example*:* Users should be alerted when they receive a new email message.

Table 1. Matrix mapping the relationship view.

Verbs / Requirements	v1	v2	v3	v4	…	vn
R1				♋		
R2	♋					
R3		♋				
R4	♋	♋	♋	♋		
…						
Rn						

Table 2. Matrix mapping the refined relationship view.

Verbs / Requirements	v1	v2	v3	v4
R4	♋	♋	♋	♋

Table 3. Identifying dominating verb.

Verb	v1	v2	v3	v4
v1		♋	♋	♋
v2				
v3				
v4				

Syntax 3 → NP—VP1—"and"—VP2:

Description: In this syntax, noun phrase precedes the verb phrase which precedes the word "and" that's finally precedes another verb phrase.

Example: Professors should be able to view student information and moderate discussions.

Rules to identify dominating verb based on syntax morphology

Based on the analysis on the identified syntax, some simple rules are derived to identify the tangled concern in a requirement shared by more than one verb.

Rule 1 → If the syntax = NP—VP1—"to"—VP2—CONJ—VP3, then VP2 is triggered by VP1 and VP3. Hence the dominating verb in this case is VP2.
Rule 2 → If the syntax = NP—VP1—CONJ—VP2, it indicates that the behavior of VP1 is triggered by VP2. The conjunction in between the two verb phrases, VP1 and VP2, provides dominating characteristic to the constituent on the left side of the conjunction, in this case, VP1. Thus VP1 dominates the syntax.
Rule 3 → If the syntax = NP—VP1—"and"—VP2, the weight for both verb phrase are equal. This is because of the conjunction "and". Hence there is no dominating verb in this syntax.

3CI DESIGN AND IMPLEMENTATION

3CI is a web-based tool that processes text-based requirements documents written in English language to identify crosscutting concerns. It is developed to materialize the crosscutting concern identification model developed in this research. The tool is meant to aid the system developers to effectively analyze the requirements document for crosscutting concern identification. The tool receives text-based requirements document as input and tags each text in the requirement phrases with the corresponding part-of-speech categories using Part of Speech Tagging (POST) module incorporated within the tool. Based on the output from POST, the tool will impose a set of rules on step-by step basis until the crosscutting concern is identified if such concern exists in the requirements document. Detailed descriptions on 3CI are discussed by Ali and Kasirun (2008a).

Use case model

The use case model depicts conceptual design of the 3CI tool system. The system is divided into three (3) modules sharing a common database. Figures 3, 4 and 5 illustrates use cases for the three modules.

Entity relationship diagram

The entity relationship diagram (ERD) in Figure 6 illustrates the interrelationship between the entities in the 3CI database.

System architecture

3CI Tool architecture is complex. The system interacts with an

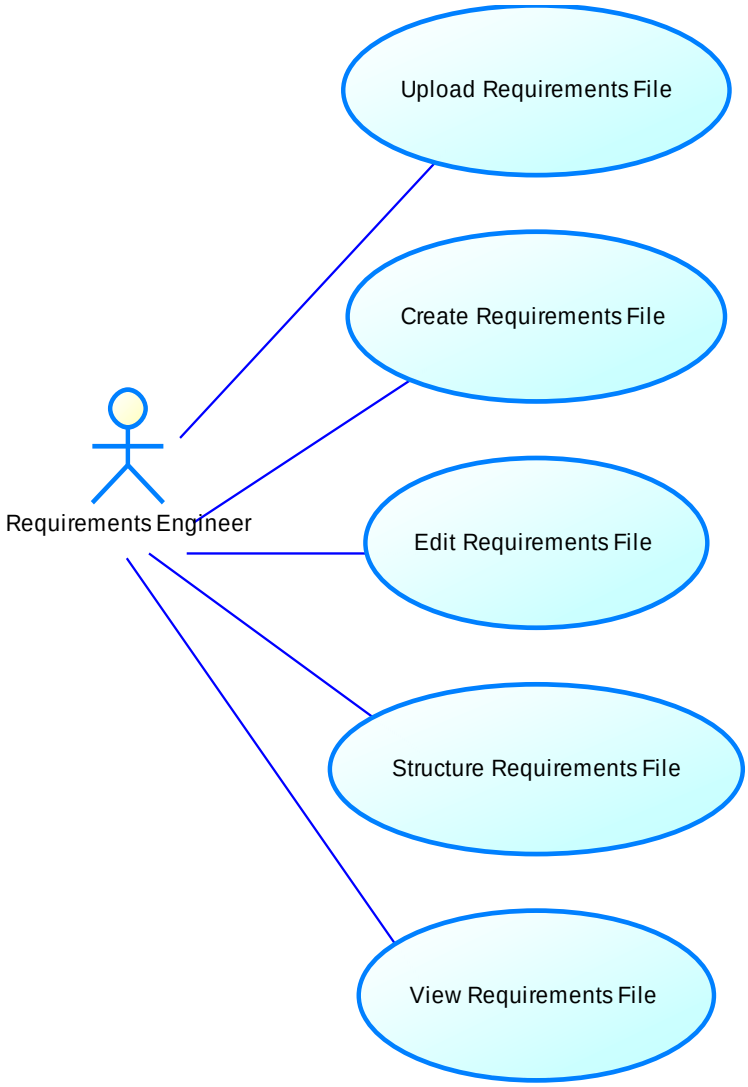

Figure 3. Use case for manage requirement document module.

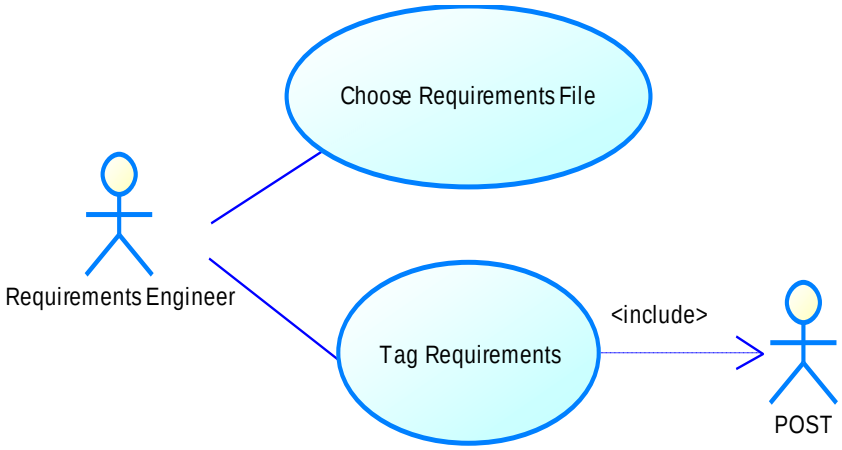

Figure 4. Use case for part of speech tagging module.

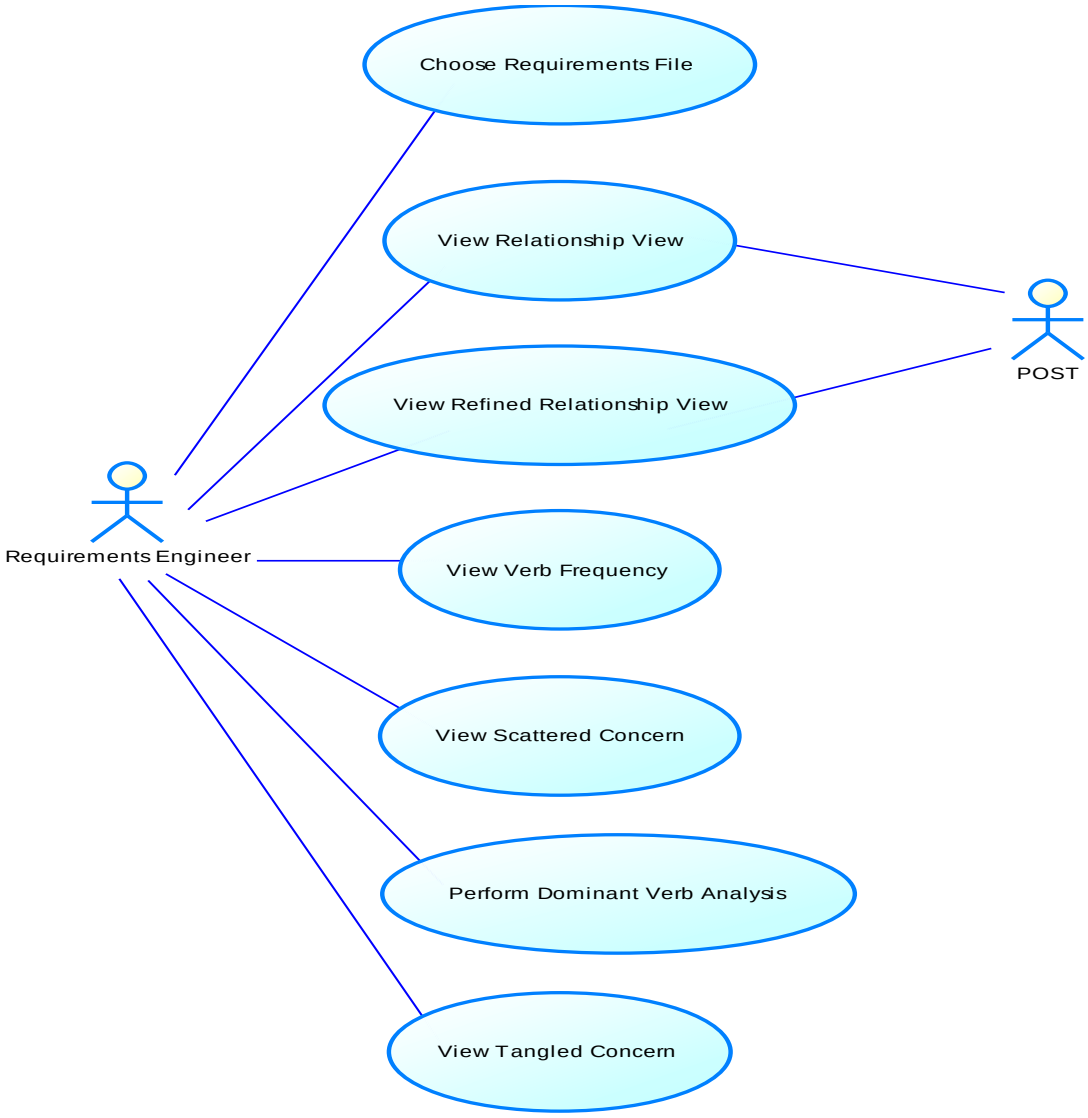

Figure 5. Use case for identify crosscutting concern module.

external C++ Application, which is the POST Tagger. Direct data binding between a PHP Application and a C++ application is not possible. Hence XML is used as an intermediate bridge between the two platforms. Figure 7 shows the tool architecture.

Working procedure

3CI consists of three (3) modules:

1. Manage requirement document module
2. Identify crosscutting concern module
3. Post tagger module

Figure 8 illustrates the sequence of operations, movements of data, decisions and storage activities of the system.

A simplistic first step in identifying crosscutting concern using the 3CI Tool is to either upload an English based text document containing requirements or create/edit the text file using Manage Document module. The next step is to structure the requirement file by assigning unique numbers to each requirement in the file and save each requirement uniquely in the database. Then the structured requirements are sent as input to the POST module to tag each word in the requirement clause. Next the tagged requirements will be saved in the database. Based on the output produced by POST, the requirement clauses having more than one (1) verb are extracted. If there is such requirement, then the system will conclude the tangled concerns. Next, dominant verb analyses are performed on all the tangled concern to identify which requirement consist dominating verb and which verb within the requirement is dominating the requirement clause. At the same time frequency analysis is performed to identify the frequency of each verb in the requirement document. Based on the analysis the verb having the maximum frequency is concluded as the scattering concern. Finally the crosscutting concern is concluded based on the output of the dominant analysis and frequency analysis.

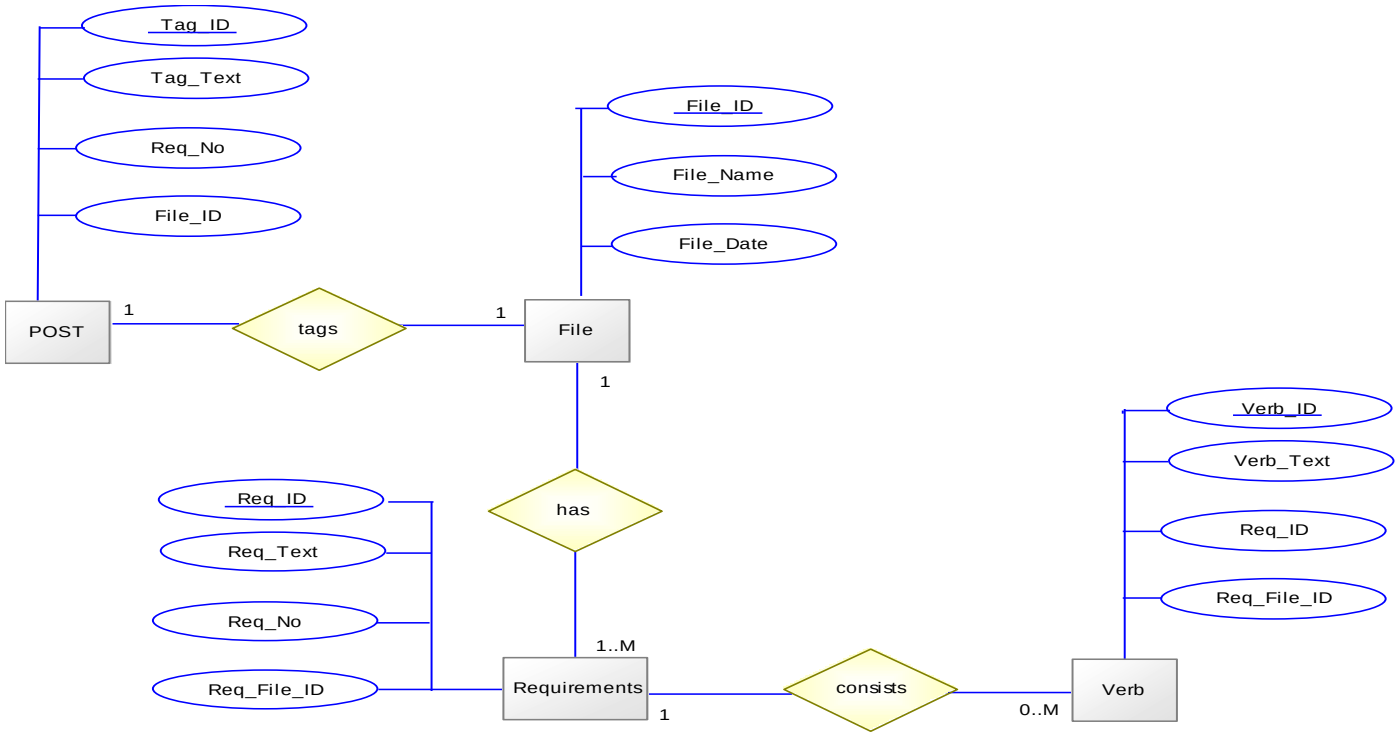

Figure 6. Entity relationship diagram.

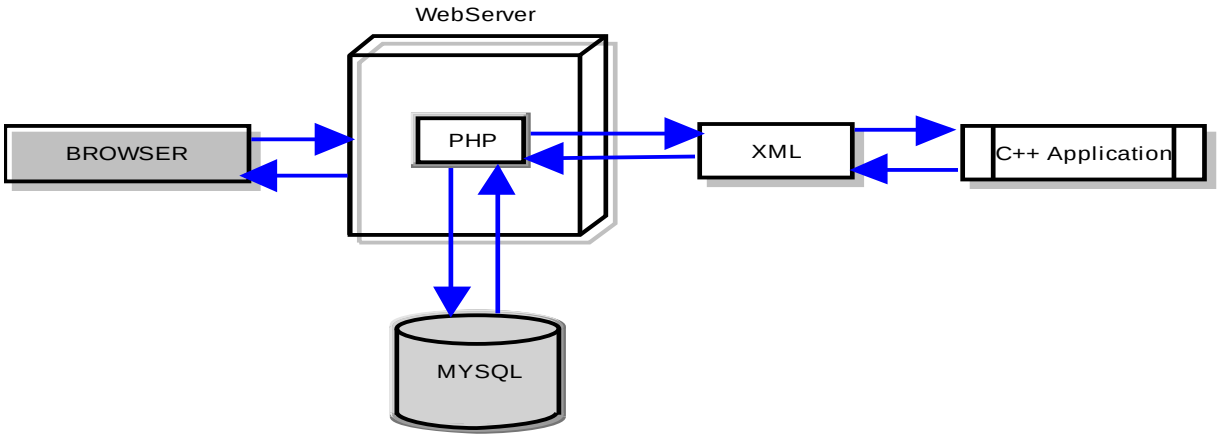

Figure 7. 3CI tool architecture.

In Figure 9, 3CI displays a raw requirements file sent as input by the user. The file contains requirements of a campus messenger system. Figure 10 shows the structured requirements to be parsed to the POST module for part of speech analysis task.

Figure 11(a) maps a relationship view based on the verbs identified in each requirement by POST. Then the matrix is refined in Figure 11(b) showing only the requirements shared by more than one verb. Figure 11(c) shows the verb frequency analysis performed on each identified verb. The result of the frequency analysis which indicates the scattering verb is shown in Figure 11(d). Figure 11(e) illustrates dominant verb analysis to identify tangled verb in each requirement clauses identified in Figure 11(b).

Algorithms

The main functions in this tool are structure requirements, verb frequency analysis and dominant verb analysis. The algorithms to perform these functions are given below.

Algorithm to structure requirements

Repeat while end of file
Read requirement paragraph while x='.'
Split sentence

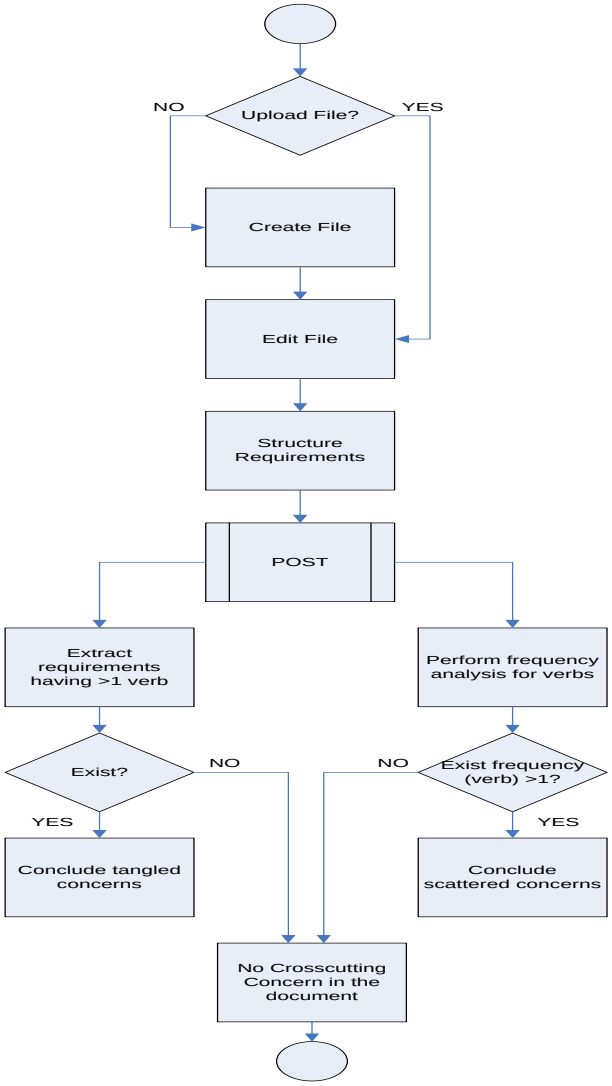

Figure 8. System flow for 3CI tool.

Write each requirements clause in a new line

Discard redundant requirement clause
Assign a unique identifier for each requirement clause

Algorithm to perform verb frequency analysis

For each requirement clause
Read unique verb phrase = i
Calculate ∑(i)
Print i, ∑(i)
Scattering verb = max [∑(i)]

Algorithms to perform dominating verb analysis

While read refined relationship view list
If syntax = NP—VP1—"to"—VP2—CONJ—VP3
then print "Dominating verb is VP2"
else if syntax = NP—VP1—CONJ—VP2
then print "Dominating verb is VP1"
else if syntax = NP—VP1—"and"—VP2
then print "No Dominating Verb"
else
print "Syntax Undefined"

Evaluation of 3CI

3CI was evaluated in terms of usability, efficiency and system scalability by conducting case studies and analyzing the required time for execution to identify crosscutting concern. Our evaluation of 3CI helps us to compare its performance with regards to manual analysis and tool-based approach. Based on the evaluation, the tool is found to be easy to use, efficient and able to process huge documents.

The evaluation framework

We compared results of 3CI with manual analysis of two case studies of varying sizes.The first case study used was Campus Messenger System described in Figure 12. This is a simple case study considering the size of the input document (120 words, 1 page). The second case study was the auction system described in Auction System Problem Description (2008). The size of this file is significantly larger than the previous case (443 words, 2 pages).

The execution of both methods (tool-based and manual) was carried out independently. The first case study, campus messenger system was used to measure the execution time for the tool and also for the manual conduct. The time was set and logged for both methods.

The scalability of the tool was measured by comparing the execution time of the two case studies based of the varying sizes of the documents. 3CI accuracy was measured by inputting several test requirement statements and comparing the actual output produced with an expected output. Table 4 tabulates the results of the accuracy test.

RESULTS OF EVALUATION

Based on the case-study, it is found that only a few steps need to be performed by the user to retrieve the output from the 3CI. Hence the tool usability is proven to be easy and user friendly.

As illustrated in Figure 13, the 3CI based analysis outperformed the manual one. In terms of required time, for the campus messenger system it was 19 times faster (78 s vs. 1463 s). Therefore the vast difference in the execution time proves that the tool-based approach is more efficient in terms of performance.

As shown in Figure 14, the larger the size of the document the longer the processing time taken to identify crosscutting concern. 3CI tool is able to accept large documents as input. With the size of 120 words it took 78 s to process Campus Messenger problem description and with the size of 443 words, it took 228 s to process the auction system problem description. In our view, this experiment demonstrates that the tool is of high potential to aid the tedious task, identifying crosscutting concern

Figure 9. Description of campus messenger system.

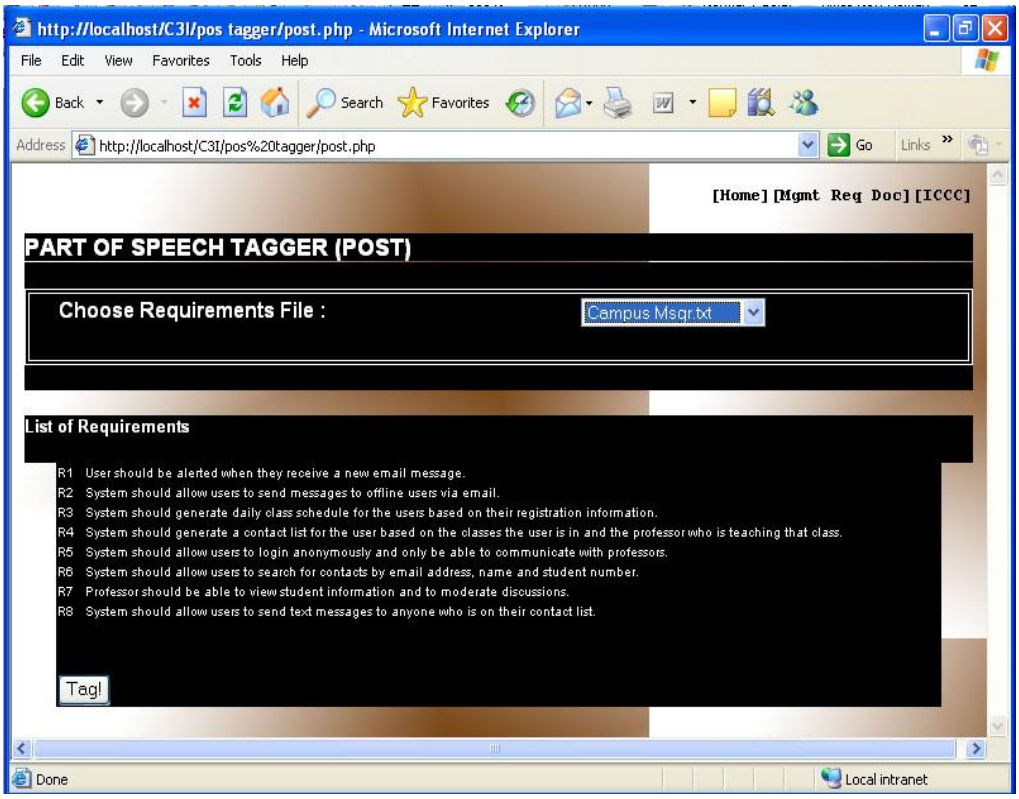

Figure 10. Structured description of campus messenger system.

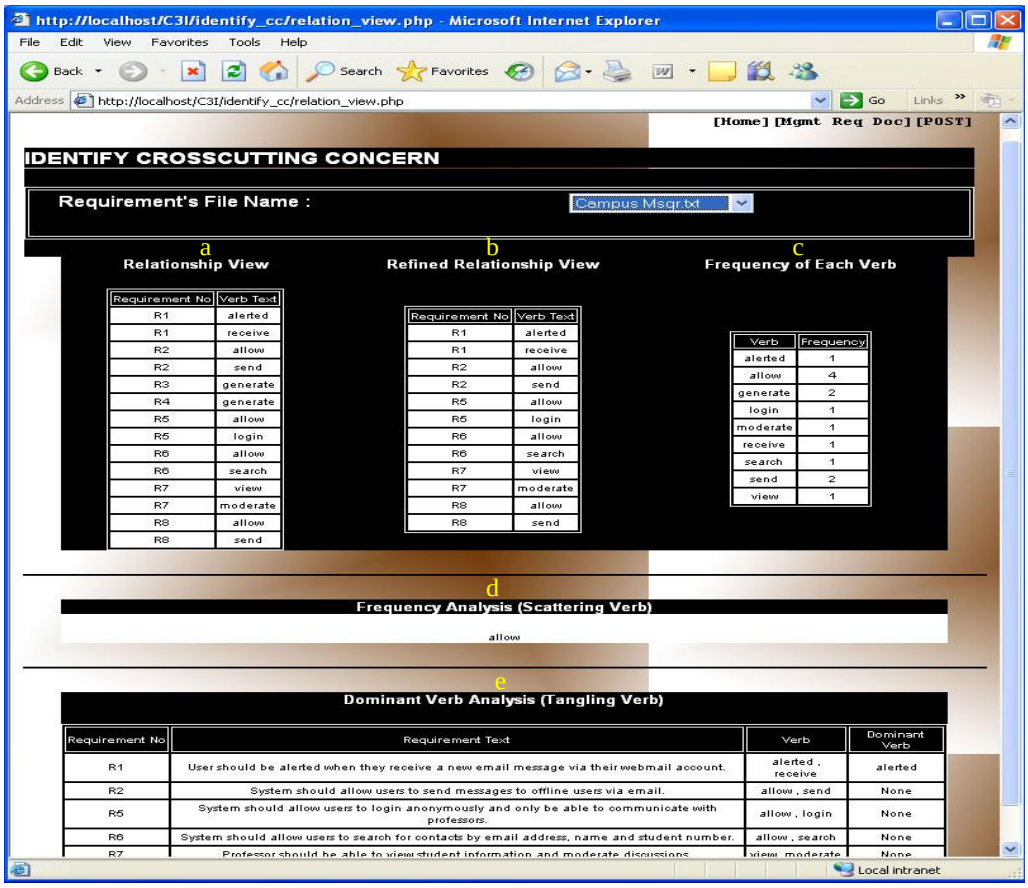

Figure 11. (a) Relationship view; (b): Refined relationship view; (c): Verb frequency; (d) Verb frequency analysis; (e): Dominant verb analysis.

Campus Messenger System Requirements

User should be alerted when they receive a new email message. System should allow users to send messages to offline users via email. System should generate daily class schedule for the users based on their registration information. System should generate a contact list for the user based on the classes the user is in and the professor who is teaching that class. System should allow users to login anonymously and only be able to communicate with professors. System should allow users to search for contacts by email address, name and student number. Professor should be able to view student information and to moderate discussions. System should allow users to send text messages to anyone who is on their contact list.

Figure 12. Requirements for campus messenger system.

at the requirements level (Figure 15).

$$\text{Accuracy} = \frac{\sum \text{Actual Output}}{\sum \text{Expected Output}} \times 100$$

$$= (¾) 100$$
$$= 75\% \quad (1)$$

Based on the evaluation using the Equation (1), the tool is found to be easy to use, efficient and able to process huge documents. The accuracy measure for the tool is 75%.

3CI AND RELATED APPROACHES

As mentioned, this model is developed based on the Theme /Doc and Ea-Miner Approaches. Theme / Doc Approach provides a semi-automated identification of crosscutting concerns in requirements specification

Table 4. Accuracy test.

	Input	Expected output		Actual output	
		Scattered crosscutting concern	Tangled crosscutting concern	Scattered crosscutting concern	Tangled crosscutting concern
1	Requirements for campus messenger system (Refer to Figure 12)	'allow'	'alerted'	'allow'	'alerted'
2	Requirements for security system (Refer to Figure 15)	'unauthorized'	'authorized'	'initiated'	'authorized'

Table 5. Comparison between relates approaches.

Approach	Theme/Doc	Early aspects identification	Proposed approach (3CI)
Mode	Semi-automated	Semi-automated	Automated
Input type	Structured document	Unstructured document	Unstructured document
Output	Action view model	View point	Matrix
Scalability	Small scale document	Any size document	Any size document
Tool	Theme/Doc Tool	EA-Miner	3CI
User intervention	Required	Required	Not Required

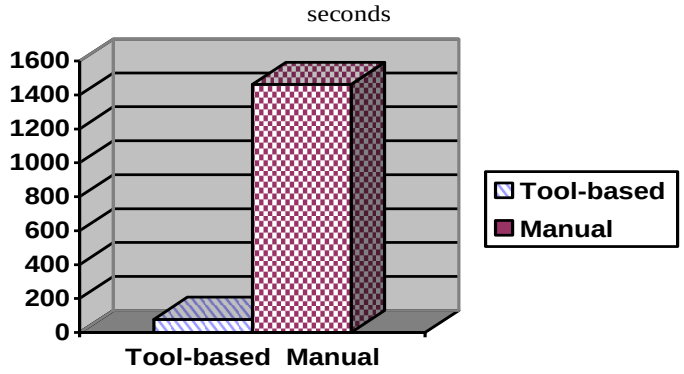

Figure 13. Comparison of time spent for tool-based and manual approach for identifying crosscutting concern.

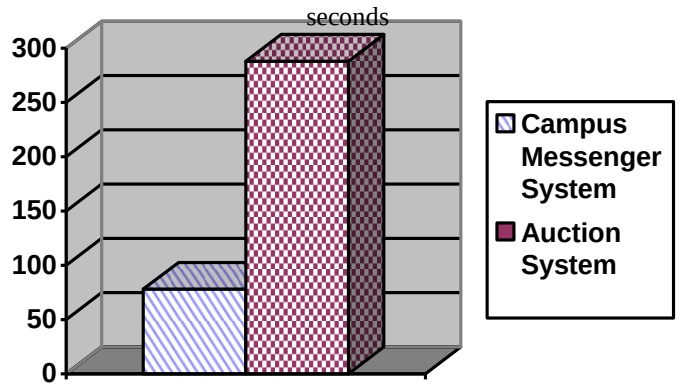

Figure 14. Comparison of varying size document.

documents. This approach is based on lexical analysis. The developer needs to go through the whole document to identify a set of action words presented as 'theme', in other words the candidate crosscutting concerns. Next the themes are fed into the action view model to generate relationships between the action words. Based on the action view model, the requirements engineer has to manually conclude on the crosscutting concerns. In order to apply this approach, the requirements engineer needs to have the domain knowledge of the system to be developed. Early aspects identification method (EA-Miner) utilizes corpus-based natural language processing (NLP) to enable identification of crosscutting concerns in a semi- automated way. This approach enables the requirements engineer to automatically mine the requirements from structured or unstructured sources to identify and build a structured aspect-oriented model of the requirements. However, the tool uses semi-automatic features for producing an intermediate model using viewpoints. Thus the tool requires the intervention of the requirements engineer for creating the models. A comparison between the proposed approach and the two approaches above are provided in Table 5.

Conclusion

This research provides a new approach for the software developers to manage and analyse requirements document effectively by presenting an automated approach to identify crosscutting concern at the

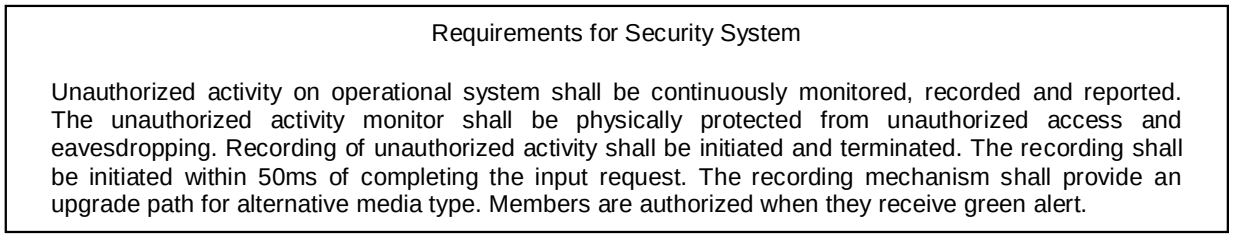

Figure 15. Requirements for security system.

requirements level and developing a tool called 3CI to aid the time consuming and error-prone activity.

The research is significant especially for analyzing requirements before finalizing them into requirements specification. Future possible work would be to integrate the tool with an elicitation tool for analyzing requirements gathered through elicitation process.

REFERENCES

Ali SB, Kasirun MZ (2008a). 3CI: A Tool for Crosscutting Concern Identification. Proceedings of International Conference on Innovation in Software Engineering (ISE).

Ali SB, Kasirun MZ (2008b). Crosscutting Concern Identification at Requirements Level. Malaysian J. Comp. Sci., (ISSN 0127-9084).

Araujo J, Moreira A, Brito I, Rashid A (2002). Aspect-Oriented Requirements with UML. Workshop on Aspect-Oriented Modelling with UML.

Auction System Problem Description. Retrieved from http://lgl.epfl.ch/research/fondue/case studies/auction/problem-description.html, September 30, 2008.

Baniassad E, Clarke S (2004). Finding Aspects in Requirements with Theme/Doc. Proceedings of Early Aspects (AOSD).

Brito I (2004). Aspects Oriented Requirements Engineering. Proceeding of the7th International Conference on Unified Modelling Language (UML) Doctoral Symposium.

Rosenhainer L (2004). Identifying Crosscutting Concerns in Requirements Specifications. Proceedings of OOPSLA Early Aspects. Aspect-Oriented Requirements Engineering and Architecture Design Workshop.

Sampaio A, Rashid A, Rayson P (2005). Early-AIM: An Approach for Identifying Aspects in Requirements. Proceedings of Requirements Engineering.

Sutton MS, Rouvellou I (2002). Modeling of Software Concerns in Cosmos. Proceedings of the 1st International Conference on Aspect-Oriented Software Development, pp. 127–133.