Journal of Theoretical and Applied Information Technology
31st March 2022. Vol.100. No 6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CROSSCUTTING CONCERNS (ASPECTS) IDENTIFICATION IN THE EARLY STAGE OF ASPECT-ORIENTED SOFTWARE DEVELOPMENT

AWS A. MAGABLEH[1], RAZAN RABABAH[2]

[1]Department of Information Systems, Faculty of Computer Science and Information Technology, Yarmouk University, Irbid, Jordan

[2]Department of Information Systems, Faculty of Computer Science and Information Technology, Yarmouk University, Irbid, Jordan
E-mail: [1]Aws.magableh@yu.edu.jo, [2]2016930005@ses.yu.edu.jo

ABSTRACT

Aspect-oriented programming (AOP) can effectively solve code-tangling and code-scattering that are caused by the attributes of crosscutting concerns or aspects. To date, research in the field of aspect orientation has concentrated on the implementation stage of the software development life cycle. Also, quite a good number of studies on aspect orientation have been conducted on the later stage of software development. However, less attention has been paid to aspect orientation at the requirement analysis stage, i.e., the early stage of software development. Therefore, the aim of this research is to propose an approach for analyzing and extracting crosscutting aspects from software requirements specifications by using a natural language processing technique, namely, the educated text stemmer algorithm. Moreover, this research focuses on the auto analysis and extraction of aspects from specifications written in the Arabic language because a lot of companies and organizations are still using English language-based requirements approaches for Arabic systems. The results of the proposed approach show promise because the proposed Arabic miner tool was able to extract the greatest number of candidate aspects from Arabic requirements specifications. The approach was evaluated by comparing the results that it automatically extracted with those extracted manually by software engineering experts. The accuracy of the experts' efforts was 88%, whereas the accuracy of the proposed approach was 36%, which can be ascribed to the poorness of the Arabic data set and Arabic WordNet tool. Nevertheless, the proposed approach has potential, and the experimental results offer insights on how to further develop the proposed tool.

Keywords: *Crosscutting concerns, Aspects, Aspect-oriented, AO, Aspect identification, Aspect elicitation, Early stage of software development life cycle, Arabic requirements specification, AO miner.*

1. INTRODUCTION

Requirement engineering (RE) is one of the most important factors that affect the success of the software development process because in requirements analysis the software analyst identifies the stakeholders' and customers' needs. One of the main approaches used by the system analyst is the object-oriented analysis and design or OOAD approach, which helps the system analyst to identify both the user requirements and the system requirements. This approach enables the identification, discovery and selection of the main business logic, as well as the various packages and classes of the system and their relationships. Generally, these relationships can be classified into primary concerns and crosscutting concerns, as illustrated in Figure 1.

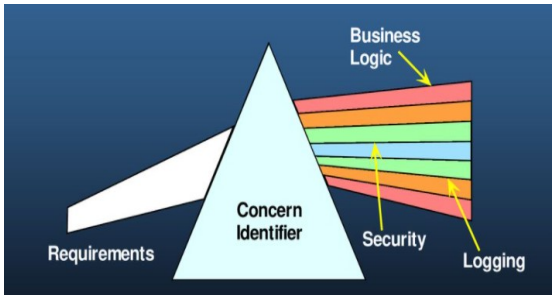

Figure 1: Requirement's classification [32]

Although an object-oriented (OO) approach is sufficient for achieving success in software engineering projects, the approach has some inherent problems that affect its ability to handle issues related to the maintainability and modularity of the developed system [23]. In particular, the OO approach is unable to deal with relatively new terminologies such as crosscutting concerns (or aspects). An aspect represents any behavior that cannot be modularized and composed effectively in one single entity or object. It might be a non-functional requirement or a specification such as reliability, security, and logging that cannot be represented by using the predefined techniques and tools available in OO approaches. Any attempt to deal with aspects as an OO object/class will lead to the poor modularization of the aspects, which will then accordingly lead mainly to two problems in the design and later stages of the software development process, namely, scattering and tangling. As illustrated in Figure 2, scattering occurs when the responsibility related to or assigned to an abstraction and localized in the abstraction is also localized at one or more other abstractions that are not responsible for said responsibility [32], while tangling occurs when an abstraction deals with responsibilities that do not belong to its scope [31] Thus, the aspect-oriented (AO) approach was invented to resolve these problems.

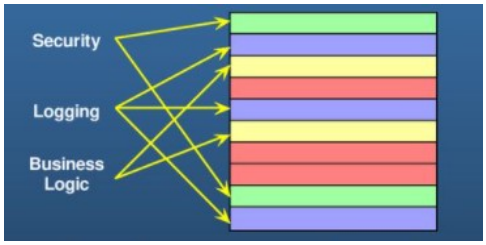

Figure 2: Scattering and tangling [32]

Therefore, the purpose of this research is to build an approach that can automatically extract aspects from software requirements specifications. The work also specifically focuses on requirements written in the Arabic language. A good analysis of requirements reduces the possibility of errors arising in the later stages of the software development project that can waste time, effort and money. For system development and the automation of software development activities, robust techniques are required to elicit aspects from natural-language text. The systems that have been developed in the Middle East suffers from the drawback that software requirements specifications have to be written in the English language in order for them to be automatically analyzed.

It is posited here that natural language processing (NLP) and machine learning (ML) techniques can be used for handling the Arabic language, thereby enabling the construction of an approach that can help requirements engineers in the requirements engineering step of the software development process who are working for clients who require Arabic language-based systems. It is envisaged that such an approach will not only facilitate the completeness and correctness of aspect elicitation, but also reduce the effort spent by requirements analysts in translating Arabic terms into the English language so that they can use existing requirements analysis approaches. In short, the motivation for developing a tool for identifying Arabic system requirements is twofold: (1) to enhance software systems that are based on the Arabic language, especially those are used by entities such as government ministries in Saudi Arabia and (2) to enhance the Arabic language usage so that it can be used in the software development life cycle (SDLC).

It has been noticed in software development that Aspects/crosscutting concerns are being attempted to be identified, modelled, and programmed at later stages to the software developments, which means there is no consistency of mapping between requirements and developments, cost of change will be high, cost of maintenance if any amendments have to take place on the object and aspect will need high effort and high cost. Additionally, Arabic community and society of programmers are increasing and Arabic system users and clients whose requirement is in Arabic are also increasing thus, it was observed that a need to have an approach that supports Aspect/crosscutting concerns identification from Arabic client requirements is essential.

2. PROBLEM STATEMENT

The requirements analysis process is critical for the success of a software development project. Recent research studies have attempted to find ways to automatically extract aspects from software requirements specification documents in order to avoid the time-consuming task of manually extracting the aspect requirements and to reduce the errors that can occur in the later stages of the SDLC. A decent number of research studies have investigated aspect-oriented (AO) requirements engineering and the development of tools for the mining of candidate aspects. However, to date, to

the best of our knowledge, there are no research studies that have investigated aspect extraction from Arabic software requirements specifications.

This research proposed an involuntary approach for obtaining aspects/Crosscutting approach from Arabic user requirements based on NLP techniques in order to share a common understanding of the structure of information among stakeholders and software engineer. The proposed approach relies on mixing of a proposed set of identification rules to identify the candidate aspects, and an automated NLP algorithm. our approach is the first to utilize an Arabic natural language algorithm to extract candidate aspects from Arabic requirements. The research was motivated by the idea that a good analysis of requirements would reduce the possibility of errors arising during the later stages of software development projects which result in wasted time, effort, and money.

3. RESEARCH QUESTIONS

This research attempts to answer the following questions:

- **RQ1:** How feasible is it to implement an aspect miner to elicit crosscutting concerns from Arabic requirements specifications that support the early stages of aspect-oriented software development (AOSD)?

- **RQ2:** How can the accuracy of an Arabic aspect miner be measured and how accurate is the miner?

4. LITERATURE REVIEW

In this section, first, some background on the concept of aspect orientation is provided in order to highlight the contribution of this study. Then, the techniques that are used to elicit aspects are reviewed.

4.1 Aspect-Oriented Software Development (AOSD)

Aspect-oriented software development [26] is based on the advanced concept of the separation of concerns. The aim of this type of software development process is to encapsulate crosscutting concerns into new units called aspects in order to enhance the modularity of the software. Four main types of AOSD process [27] can be employed in the SDLC. The first is aspect-oriented requirement engineering (AORE), which is used in the early stage of AOSD. Aspect-oriented requirement engineering enables the requirements engineer to identify, specify, represent, and document crosscutting concerns (aspects) by providing the engineer with the systematic means and methodologies to do so. Various AORE approaches [28] have been developed to deal with crosscutting concerns, based on different perspectives, and these approaches deal with functional concerns, non-functional concerns, or both. However, all the approaches share the same goals, which are aspect identification, aspect separation, aspect representation, and aspect composition [8][9][10].

The second type of AOSD process is aspect-oriented modeling or AOM. This process is implemented in the design stage of the SDLC, when the designer has to realize the proposed system, which contains a set of requirements. At this point in the software development process, the designer has to think about the behavioral aspects of the system as well as its structural aspects.

A third process that can also be applied in AOSD is aspect-oriented programming (AOP). Aspect-oriented programming is a relatively new programming paradigm that has been introduced with the objective of creating better and more effective systems. In order to understand how AOP works, it is important to know the following terminology:

- Join point: A well-defined point in program implementation (calling or executing techniques are examples of join points);
- Point-cut: A collection of models used to select a join point;
- Advice: A method-like structure containing extra behavior to be added to and injected into the corresponding join point;
- Aspect: An embodiment of a crosscutting concern (in the context of OO programming, an aspect is somewhat comparable to a class);
- Aspect weaving: A method by which the aspect's behavior is combined with the initial code – in other words, weaving is a method of combining the crosscutting concerns with a program.

The fourth and final AOSD process is aspect-oriented quality assurance or AOQA. This is used in the testing phase and is a significant part of the design process. Detecting software faults is one of the key objectives of testing. Such testing takes place throughout the SDLC because it is crucial

that defects are identified as soon as possible during the development of software systems.

Any comparison of software development approaches needs to be done at the phase level of the SLDC because each phase of the SDLC has its own requirements. Hence, there are different comparison criteria for each phase. In the AOSD approach, the criteria for the requirement engineering phase is based on the provision of support for the aspectual concerns at the early stage of software development, and there should be distinct mapping between the aspects and the final products or goals of the system. The criteria that are used for the software specification phases are based on the extent of the usage of the supporting tools, the verification steps, and the independency of the language. In the software design phase, the criteria that are used are the quality properties and the extent of the traceability and the reusability of the Aspects [4].

4.2 Aspect-Oriented Requirement Engineering (AORE)

The inability of OO to represent and decompose crosscutting concerns at the early stage of software development led to the development of AORE [2]. Aspect-oriented requirement engineering aims at achieving the advanced separation of concerns. It mainly involves the use of modular reasoning and composition realization. Modular reasoning is related to keeping the concern isolated from any other concerns that are affected by or influenced by the concern in order to understand the concern in isolation. Composition realization realizes the dependencies and the relationships between concerns in order to understand the properties of the system as a whole [20][21].

Aspects or crosscutting concerns are entities that represent the requirements that are not related strictly to specific decomposition modularization units. For instance, the use case unit is the best option for the modularization of functional requirements. However, non-functional requirements [30] cannot belong to a specific use case module. Instead, such requirements normally crosscut multiple use cases. This problem is solved by the use of the crosscutting concern or aspect, which acts as an encapsulation of one or more requirements related to a specific matter of interest [11]. Hence it can be said that the crosscutting concerns concept denotes the relationship between two elements when one of those elements affects the other element or restricts the other element's implementation [18].

In 2008, Blaine and colleagues stated that there are some issues that need to be addressed in regard to how to elicit quality requirements, how to find the trade-off between them, and how to measure the quality requirements [5]. A decade later, in 2018, Tamai and colleagues tried to solve these problems by proposing a tool named the QR miner [10] for analyzing the quality requirements in the software requirements specification in terms of their volume, balance, and structure. To achieve this goal, they used ML and NLP, and especially a deep learning technique to find and classify quality requirement sentences. Figure 3 below shows the flow of the QR miner.

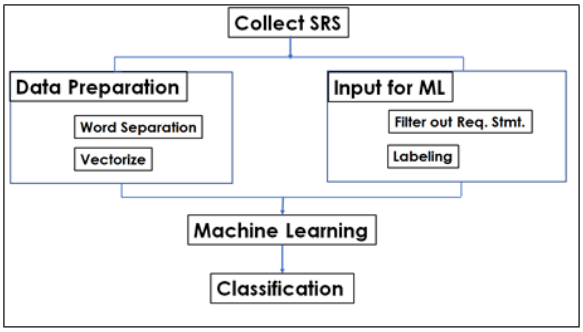

Figure 3: QR miner [10]

Earlier, in 2005, Sampaio and his team proposed the EA-miner tool [27] to automatically extract the aspects and their crosscutting relationships, and the non-aspectual requirements by utilizing the WMATRIX [29] natural language processor that pre-processes the input document to get relevant information. The WMATRIX provides part-of-speech (POS) and semantic tagging, as well as frequency analysis. The flow of the EA-miner is illustrated in Figure 4 below.

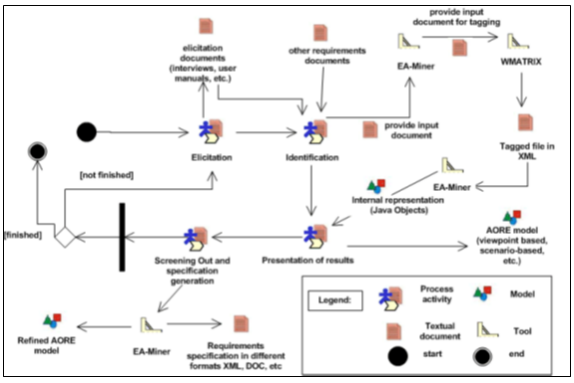

Figure 4: EA-miner approach [29]

Later, in 2010, Boudlal and his team proposed a tool that can be used to perform multiple NLP tasks on the Arabic language, such as morphological analysis, disambiguation and (POS) tagging. Their work is an extended version of the Alkhalil Morpho Sys analyzer [1] which is based on a large set of Arabic morphological rules and on the integration of linguistic resources [6]. Microsoft has developed by Microsoft Advanced Technology Lab in Cairo for processing Arabic text. It is a set of NLP components, such as a morphological analyzer (SARF), parser, spell-checker, auto corrector, named entity recognizer (NER), POS tagger, etc. These components are presented to the users in the form of web services.

In a similar vein, in 2009, Habash and his team proposed a toolkit that can be used to process Arabic natural language. It performs different tasks such as POS tagging, discretization, etc. In the toolkit, the morphological analysis and disambiguation for Arabic (MADA) component handles the morphological analysis and disambiguation of Arabic text, and the (TOKAN) component is used to define a large set of possible tokenization schemes that can be generated from the disambiguation analyses produced by the MADA component. The researchers proposed a semi-automated approach for constructing a sequence diagram from Arabic-language software requirements, which is based on the use of a MADA+TOKAN Arabic parser as a first step, which parses each token in the text and associates it with its corresponding tag. After the parsing step is completed, and before moving to the second phase, the tags are placed into three groups: subjects, receivers, and participants. The subject group must be defined in the parsing phase because the subject group defines the caller or the sender in the sequence diagram. On the other hand, the receiver and participant must be defined during the construction of the sequence diagram phase. Hence sequence tables also need to be constructed during the analysis process [13].

Other approaches that have been developed to solve the identification of requirements include that [14] [15] [17], who proposed a method for classifying requirements into functional, non-functional, temporal, and non-temporal by using the J48 and the PERTree classification algorithms. Li in [33] presented an example of the usage of deep learning in one of the software engineering tasks, namely, bug report summarization. The researchers found that there is a noticeable trend of using deep learning techniques in different software engineering phases. However, they also found that there are effectiveness, efficiency, understandability, and testability problems in the usage of deep learning models in software engineering task [16] classified the content of the software requirements specification into requirements or information by using convolutional neural networks. On the other hand, [25] proposed a methodology to evaluate the quality of system requirements as the quality experts using a ML algorithm (the rule induction algorithm). Other researchers have tried to analyze AO requirement using formal methods [7] [3].

In the current study, the algorithm that is used for the identification of crosscutting concerns (aspects) in the early stage of AOSD is the (ETS) algorithm (Al Shammari & Lin, 2008). The ETS algorithm is an Arabic stemmer. The reason behind the usage of this stemmer is that the other available Arabic stemmers suffer from high stemming error rates. Generally, Arabic stemmers stem all the words blindly and show poor performance in respect of compound words.

Therefore, in the first phase of the proposed approach, the ETS algorithm is used to preprocess the Arabic requirements text. It is also used to apply a part of one of the proposed heuristic rules that is employed to improve requirements identification. The second phase involves the identification of the participants, where the participant groups include the sender/caller, the main actor, and the receiver. Then, the next phase is message identification, and usually the message is the action for each statement. The action here is defined as the object. To the best of our knowledge, the proposed approach is the first to be built to extract candidate aspect requirements in the Arabic language.

The literature did not focus on obtaining Aspects from Arabic requirement, it was always focused on obtaining and identifying Aspect from English requirement text, thus not much has been done on Arabic language.

5. RESEARCH METHODOLOGY AND PROPOSED APPROACH

5.1 Overall Research Design

This research proposes the usage of NLP techniques to build an approach to extract a list of candidate aspects from Arabic software requirements specifications. Figure 5 illustrates the steps of the proposed approach, which are described in detail in the following.

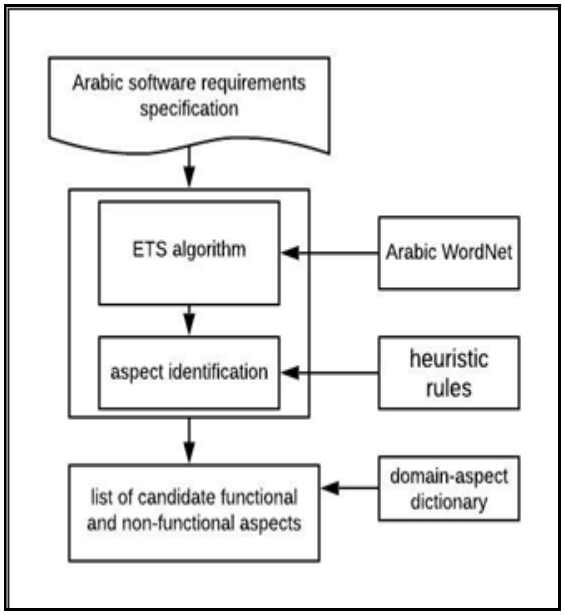

Figure 5: Proposed aspect extraction approach

The proposed approach accepts an Arabic software requirements specification and an Excel file that contain most of the aspects that are used in different system domains. These aspects have been translated into the Arabic language to create a domain aspect dictionary as input text especially for this research.

In the proposed approach, first, the non-functional aspects are extracted. Then, the proposed algorithm and Arabic WordNet are used to extract the functional aspects. This extraction process begins with normalization, which is followed by stop word removal and tokenization. Then, a dictionary of verbs is automatically created, and aggressive stemming of the verbs in the documents is performed, the output of which is a list of stemmed verbs. After that, the functional aspects are extracted by finding the repeated verbs in the data set and then comparing them with the domain aspect dictionary built specifically for the purpose of this research.

5.2 Research Phases

5.2.1 Data Set

The data set used in this research is a collection of software requirements specifications that are written in the Arabic language. These specifications constitute a raw natural language text without any classification of the use cases and without any identification numbers for the requirements. The Arabic requirements of real systems are not fully available on the internet and the Arabic software requirements specification documents that are available on the internet are generally of poor quality. Therefore, in this research the used data is an English library system translated into Arabic. Hence the domain aspect dictionary is built based on English software requirements specifications [12].

5.2.2 Construction of the Functional and Non-Functional Domain Aspects Dictionary

This research uses the pure data set to extract the candidate functional and non-functional domain aspects. This is done manually by revising the documents and trying to extract the aspects based on an analysis of the system requirements and find the crosscutting concerns of the system. Then, these aspects are translated into the Arabic language and stored in an Excel spreadsheet.

5.2.3 Aspect Extraction

Before applying the ETS algorithm, preprocessing of the library system is conducted. Preprocessing means that the research approach takes the text data as input written in Arabic, which represents user requirements (a set of reconstruction rules is applied). A set of reconstruction rules is applied to transform the input user requirements into a simple, short and unambiguous format, which decreases ambiguity of the input, handles incompleteness, and produces more accurate results. After the user enters the requirements text, the syntactic reconstruction rules are applied to the input text. These rules are as follows:

- The sentences of the user requirements text should be complete, i.e., end with a full stop and have no ellipses.
- The user requirements should be written in Modern Standard Arabic or MSA.
- The user requirements should be free of misspellings and typos [24].

- The user requirements should be grammatically correct [24].
- The user requirements text should consist of short sentences.
- If a sentence is in the passive voice, it should be converted into the active voice [22].
- The user requirements should not contain hidden pronouns.

The ETS algorithm and Arabic WordNet are used to extract the candidate aspects. The algorithm depends on the usage of stop words in order to define the list of verbs and nouns in the text. Usually, stop words are removed in the early stage of text preprocessing. To identify the candidate aspects, the ETS algorithm refers to a list of stop words, in which the stop words are categorized as either useful or useless. The useless stop words do not give any benefit to the subsequent words, whereas the useful stop words can indicate the syntactical category of the subsequent stop words. Tables 1 and 2 give some examples of useful Arabic stop words.

Table 1: Arabic Prepositions Preceding Verbs

Preposition	English equivalent
حيثما	Wherever
كلما	Whenever
إذا	If
عندما	When (not for question)

Table 2: Arabic Circumstantial Nouns Indicating Time and Place

Preposition	English equivalent
إلى	until, near, towards, to
أمام	in front of
باتجاه	In the direction of
بجانب	Aside, next to, beside
بعد	After
بين	Between
تحت	Below, beneath, down
حتى	Till (time and location)
خارج	Outside of
خلال	Through, during,
عبر	Through
على	Over
فوق	Above, up
في	In (time, location, duration)
قبل	Before
قريب	Near
منذ	Since
وراء	Behind, beyond

Before applying the ETS algorithm, normalization is performed to make the data set more consistent. Normalization consists of the following steps:

- Convert text to Unicode
- Remove diacritics and punctuation
- Remove non-letters (e.g., numbers)
- Replace آ with double alif (ا)
- Replace ى with ا
- Replace initial إ with أ
- Replace all hamza forms ء, ؤ, ئ with أ.

After that, the steps work on documenting normalization and special stop words removal process. In this phase, useless stop words are removed in order to reduce the size of the data set. Then, the nouns are identified by locating stop words that precede nouns, or words starting with definite articles. The identified words are flagged as nouns in preparation for the stemming phase. In the stemming phase (first phase), the verbs have to be identified by locating the stop words that always precede the verbs. Then, the identified nouns and verbs are separately added to the global dictionary and tagged as nouns and verbs. In the aspect extraction phase, there is no need to know the identified nouns, so the noun dictionary is discarded. In the Arabic language there are no consecutive verbs, so the word that follows a verb is categorized as a noun or a stop word. If the word is not a stop word, then the word is flagged as a noun and added to the noun dictionary. Verbs are stemmed using the Khoja root-based stemmer [19], and nouns are lightly stemmed. The Khoja root-based stemmer [19] is an algorithm that works based on the elimination of the suffixes, prefixes, and infixes of a given Arabic word.

6. RESULTS AND EVALUATION

In order to extract the candidate aspects of this system, the proposed approach applies the following steps:

- The ETS algorithm and Arabic WordNet is used to extract the sentences that have verbs from the data set in order to extract all the verbs that act as the functional requirements of the system. Note the algorithm finds the functional and non-functional sentences due to the small size of the used data set.
- The ETS algorithm is used to identify the verb sentences (see Figure 6 for examples).


يستلم مدير الطلبات اشعارًا
مدير الطلبات يقوم بعمليات البحث عن مزودي هذه الكتب
يقوم مدير الطلبات بانشاء اتفاقيه مع مزودي الكتب
يقوم مزود الكتب بطلب التكلفة
يقوم مدير الطلبات بدفع التكلفة
يقوم مدير الطلبات باستقبال الكتب ترتيبها
يقوم مدير الطلبات بتسليم الكتب الى قسم التصنيف

كاشف التصنيف يضيف الكتاب الى النظام
يستلم كاشف التصنيف الكتاب وتصنيفه من خلالموظف التصنيف
يقوم كاشف التصنيف بمراجعه التصنيف والكتاب
يقوم كاشف التصنيف باضافه الكتاب الى النظام والحاله تكون قيد المراجعة
يرسل كاشف التصنيف اشعارًا الى مساعد امين المكتبه بوجود كتاب جديد متوفر
يقوم كاشف التصنيف بارسال الكتاب الى مساعد امين المكتبه لمراجعته

يتواصل مع الاعضاء ملاحظه مساعد امين المكتبه ان هناك كتاب او بعض الكتب فقدت
يبحث مساعد امين المكتبه في النظام عن العضو مستعير الكتب
يجد مساعد امين المكتبه الاسم وبيانات الاتصال بالعضو مستعير الكتب
يفرض مساعد امين المكتبه غرامه مالية
يتواصل مساعد امين المكتبه مع الاعضاء مستعيري الكتب

بحث الاعضاء عن كتاب
يستخدم العضو الكمبيوتر العام
يبحث الاعضاء عن كتاب محدد من خلال الاسم او المولف او المزود

Figure 6: ETS verb sentences

- Heuristic rules are used to find the "non-functional" requirements which act as an adjective in Arabic so the algorithm extracts them. Note that in this research, the non-functional aspects are extracted by comparing the text of the data set with the Excel file built for the domain functional and non-functional aspects (see Figure 7).


الكفاءه
الموثوقيه
سهولة الاستخدام

Figure 7: Candidate non-functional aspects

- The first heuristic rule identifies the actions (i.e., verbs) in the requirements and the actions or verbs that are identified as shared between more than one actor of the system are considered to be functional aspects (see Figure 8).

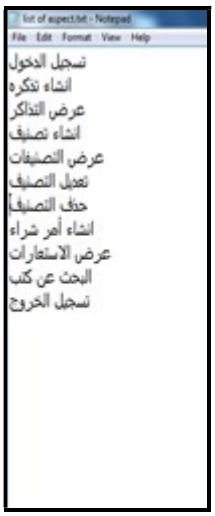
تسجيل الدخول
انشاء تذكره
عرض التذاكر
انشاء تصنيف
عرض التصنيفات
تعديل التصنيف
حذف التصنيف
انشاء أمر شراء
عرض الاستعارات
البحث عن كتب
تسجيل الخروج

Figure 8: Candidate functional aspects

The evaluation of the performance of the proposed approach was a bit challenging because, as explained earlier, it is difficult to find subject matter experts and/or respondents to help in the evaluation as aspect orientation is not well known and well used in the software development industry in the Middle East. We therefore decided to use a simple and decent evaluation method that involved comparing the aspects identified in our proof of the concept tool with the aspects extracted manually by a requirements analyst and then calculating the recall of both results. As a result of the poor availability in the local area of system analysts who understand the aspect requirements concepts, the evaluation of the research depended on only a few participants. The participants consisted of a few instructors from Yarmouk University and Al-Albayt University, a few Masters students undertaking research in the field of OO and AO in computer information systems, and two members of the local Jordanian software engineering industry. To illustrate the benefits of using the proposed automatic Arabic aspect extraction approach over manual extraction, the e-library management system requirements were used as a case study.

The results of our experiment revealed that there is a lack in the performance of the proposed approach in terms of its ability to extract the candidate aspects. This lack may be caused by the poorness of the Arabic data set, the poorness of the Arabic WordNet which does not have enough synonyms for the extracted verbs, and the poorness of the predefined dictionary of domain aspects.

Table 3 shows the number of candidate aspects that were extracted correctly and incorrectly from the user requirements. Table 4 shows the number of candidate aspects that were extracted correctly and incorrectly by the proposed approach as compared against the average number extracted by the evaluators.

Table 3: Candidate Aspects Extracted by the Evaluators

Evaluator	No. of functional aspects	No. of incorrect functional aspects	No. of non-functional aspects	No. of incorrect non-functional aspects
Evaluator 1	2	0	1	2
Evaluator 2	3	0	4	0
Evaluator 3	2	0	4	0

Table 4: Number of Extracted Candidate Functional and Non-Functional Aspects

Evaluator	# correct	# incorrect	# missing
Arabic Aspect Miner (Our Proposed Tool)	14	0	2
Evaluators (averaged)	8	1	3

When the proposed approach was applied to the same requirements, the number of extracted candidate functional aspects was 11 and the number of extracted non-functional aspects was 4. Hence the total number of extracted aspects was 14. However, the results extracted by evaluators are 8. Our proposed approach and the associated proof of concept tool managed to extract more Aspects from the same requirement used in both cases. The results summarized in Table 4.

The performance of the proposed approach was evaluated by calculating the recall, precision, and f-measure values, which are used to determine the overall level of accuracy. The precision, recall, and f-measure are calculated as follows:

$$\text{Precision} = \frac{\sum_{c \in C} |N_{correct}|}{\sum_{c \in C} |N_{correct}| + |N_{incorrect}|}$$

$$\text{Recall} = \frac{\sum_{c \in C} |N_{correct}|}{\sum_{c \in C} |N_{correct}| + |N_{missing}|}$$

$$\text{F-measure} = \frac{2 \times (precision \times recall)}{precision + recall}$$

Where the number of correctly identified elements is denoted as | N correct |, the number of incorrectly identified elements is denoted as | N incorrect |, and the number of elements that were not identified but should have been is denoted as | N missing |.

The overall performance of the evaluators in terms of precision, recall, and f-measure was 92%, 51%, and 66%, respectively. The performance of the proposed approach was as follows: precision = 45%, recall = 31%, and f-measure = 36%.

7. CONCLUSION AND FUTURE WORK

This research has some limitations. First, the scope of this study is limited by the Arabic software requirements specification documents that are available over the web. Second, the lack of availability of other research studies related to Arabic AORE limits the comparison of the results of this study with those of related works. Third, Arabic WordNet does not contain a huge set of words and synonyms, so this reduces the efficiency of the results produced by the proposed approach. Lastly, the results of the performance comparison were limited due to the lack of Arabic system analysts with expert knowledge and experience in regard to the meaning and usage of aspects because aspect orientation is not a very well-known concept in the market.

This research proposed an automatic approach for extracting candidate aspects from Arabic user requirements based on NLP techniques in order to share a common understanding of the structure of information among stakeholders and software engineer. The proposed approach relies on combining of a proposed set of identification rules to identify the candidate aspects, and an automated NLP algorithm. This approach is the first to utilize an Arabic natural language algorithm to extract candidate aspects from Arabic requirements. The purpose of this research proposal was to build an approach that can automatically extract the aspects from Arabic software requirements specifications in

order to facilitate requirements analysis by reducing the time and effort consumed in this stage of the SDLC. It was envisaged that the approach could be more accurate than those obtained by requirements engineers depending on their knowledge alone. The research was motivated by the idea that a good analysis of requirements would reduce the possibility of errors arising during the later stages of software development projects which result in wasted time, effort, and money.

Although the results of this research were not as good as hoped, they show that the proposed approach has some potential. Therefore, in future work, we aim to 1) enhance the used data set so that it is bigger and contains pure Arabic user requirements that are not translated, 2) try to enrich the Arabic WordNet with more verbs and more synonyms, and 3) study the grammatical structure of the Arabic language in order to enhance the aspect identification rules to enable the accurate extraction of more aspects.

REFERENCES

[1] Alami, N., Arman, N., & Khamyseh, F. (2017). A semi-automated approach for generating sequence diagrams from Arabic user requirements using a natural language processing tool. *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*,309–314.

[2] Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., ... & Rummler, A. (2008, September). An exploratory study of information retrieval techniques in domain analysis. In *2008 12th International Software Product Line Conference* (pp. 67-76). IEEE.

[3] Alshareef, S. F., Maatuk, A. M., Abdelaziz, T. M., & Hagal, M. (2020, September). Validation Framework for Aspectual Requirements Engineering (ValFAR). In *Proceedings of the 6th International Conference on Engineering & MIS 2020* (pp. 1-7)

[4] Braude, E. J., & Bernstein, M. E. (2016). *Software engineering: modern approaches*. Waveland Press.

[5] Blaine, J. D., & Cleland-Huang, J. (2008). Software quality requirements: How to balance competing priorities. *IEEE Software*, *25*(2), 22-24.

[6] Boudlal, A., Lakhouaja, A., Mazroui, A., Meziane, A., Bebah, M. O., & Shoul, M. (2010). Alkhalil Morpho Sys: A Morphosyntactic analysis system for Arabic texts. International Arab conference on information technology, 1-6.

[7] Can Qu , Xuhui Zhang , Haihua Chen and Lichen Zhang. (2021). Aspect-oriented requirement analysis based on Formal method, *Journal of Phys.: Conf. Ser.* 1952 042027

[8] Chitchyan, R., Rashid, A., Rayson, P., & Waters, R. (2007, March). Semantics-based composition for aspect-oriented requirements engineering. In *Proceedings of the 6th international conference on Aspect-oriented software development* (pp. 36-48). ACM.

[9] Chitchyan, R., Sampaio, A., Rashid, A., & Rayson, P. (2006). A tool suite for aspect-oriented requirements engineering. *Proceedings - International Conference on Software Engineering*, 19–25.

[10] Chitchyan, R., Rashid, A., Sawyer, P., Garcia, A., Alarcon, M. P., Bakker, J., ... & Jackson, A. (2015). Survey of aspect-oriented analysis and design approaches.

[11] Clarke, S. (2004). Theme: an approach for aspect-oriented analysis and design.

[12] Ferrari, A., Spagnolo, G. O., & Gnesi, S. (2017, September). Pure: A dataset of public requirements documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE)* (pp. 502-505). IEEE.

[13] Habash, N., Rambow, O., & Roth, R. (2009). MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, (41), 102–109.

[14] Hayes, Jane Huffman, Li, W., & Rahimi, M. (2014). Weka meets TraceLab: Toward convenient classification: Machine learning for requirements engineering problems: A position paper. *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering, AIRE 2014 - Proceedings*, 9–12.

[15] Hayes, J. H., Dekhtyar, A., & Osborne, J. (2003). Improving requirements tracing via information retrieval. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2003-Janua*, 138–147.

[16] Huaming, L., Xuehui, B., Xiuyou, W., & Weilan, W. (2017). Automatic classification of Thangka headdresses based on convolutional depth neural networks. *ACM International Conference Proceeding Series*, 105–110.

[17] Ibrahim, M., & Ahmad, R. (2010). Class diagram extraction from textual requirements using natural language processing (NLP) techniques. *2nd International Conference on Computer Research and Development, ICCRD 2010*, 200–204.

[18] Kurdi, H. A. (2013). Review on Aspect Oriented Programming, 4(9), 22–27.

[19] Khoja, S., & Garside, R. (1999). Stemming arabic text. *Lancaster, UK, Computing Department, Lancaster University*.

[20] Moreira, A., Chitchyan, R., Araújo, J., & Rashid, A. (Eds.). (2013). *Aspect-oriented requirements engineering*. Berlin, Heidelberg, Germany: Springer.

[21] Moreira, A., Araújo, J., & Rashid, A. (2005, June). A concern-oriented requirements engineering model. In *International Conference on Advanced Information Systems Engineering* (pp. 293-308). Springer, Berlin, Heidelberg.

[22] More, P., & Phalnikar, R. (2012). Generating UML Diagrams from Natural Language Specifications. International Journal of Applied Information Systems (IJAIS), 1, 19-23.

[23] Mylopoulos, J., Chung, L., & Yu, E. (1999). From object-oriented to goal-oriented requirements analysis, *42*(1), 31–37.

[24] Nassar, I. N., & Khamayseh, F. T. (2015). Constructing activity diagrams from Arabic user requirements using Natural Language Processing tool. *2015 6th International Conference on Information and Communication Systems, ICICS 2015*, (APRIL), 50–54.

[25] Parra, E., Dimou, C., Llorens, J., Moreno, V., & Fraga, A. (2015). A methodology for the classification of quality of requirements using machine learning techniques. *Information and Software Technology*, *67*, 180–195.

[26] Rashid, A., Moreira, A., & Araújo, J. (2003, March). Modularization and composition of aspectual requirements. In *Proceedings of the 2nd international conference on Aspect-oriented software development* (pp. 11-20).

[27] Sampaio, A., Chitchyan, R., Rashid, A., & Rayson, P. (2005). EA-Miner: A tool for automating aspect-oriented requirements identification. *20th IEEE/ACM International Conference on Automated Software Engineering, ASE 2005*.

[28] Sampaio, Américo, Chitchyan, R., Rashid, A., & Rayson, P. (2005). EA-Miner: A tool for automating aspect-oriented requirements identification. *20th IEEE/ACM International Conference on Automated Software Engineering, ASE 2005*, 352–355.

[29] Sawyer, P., Rayson, P., & Garside, R. (2002). REVERE: support for requirements synthesis from documents. *Information Systems Frontiers*, *4*(3), 343-353.

[30] Shah, U. S., Patel, S. J., & Jinwala, D. C. (2016). Specification of non-functional requirements: A hybrid approach. *CEUR Workshop Proceedings*, *1564*.

[31] Sutton, S. M. (2005). Aspect-Oriented Software Development and Software Process, 177–191. https://doi.org/10.1007/11608035_17.

[32] Laddad, R. (2003). Aspect-oriented programming will improve quality. IEEE software, 20(6), 90-91.

[33] Li, X., Jiang, H., Ren, Z., Li, G., & Zhang, J. (2018). Deep Learning in Software Engineering, 1–10. Retrieved from http://arxiv.org/abs/1805.04825.