A Natural Language Processing Based Model for Quality Assurance of Software Requirements Document

Bolanle F.Oladejo, Tosin Esther Olorunnishola
oladejobola2002@yahoo.com66tosin@gmail.com
University of Ibadan, Ibadan Nigeria
Computer Science Department

Abstract– Software projects failure has been a reoccurring problem in software development domain. The advent of applying engineering principles to software development initiated the need to produce software products systematically to solve the problems of poor software quality, late deliveries, and cost overruns. Requirement engineering is the first phase in software development process and the major output of this phase is the Software requirements document. This document usually written in natural language contains both functional and non-functional system requirements. However, due to the informal nature of natural language, it is prone to a lot of errors, which if not detected early enough has an adverse effect on the success of the software project. Hence, this work aimed at the development of an NLP based model for quality assurance of Software requirements. Using two major techniques which are Natural Language Processing and information extraction techniques, the developed NLP model was able to assure the quality of software requirements based on testability metrics. This was done by performing syntactic and semantic analysis on requirements. A Software Requirement quality assurance system was developed using JAPE grammar rules to extract thematic roles and detect non-testability features of software requirements. We report on the performance of our model using datasets of Promise data repository.

Keywords — Requirement engineering, Natural Language Processing, Software Requirements document, Software Requirement Quality assurance, Testability metrics.

1 INTRODUCTION

The fast growing technological world has given rise to the development of sophisticated software products. The traditional way of software development has been faced with the challenges of not meeting end users requirements, budget constraints, late deliveries and poor software quality [1]. These challenges initiated the need to apply engineering principles to software development.Hence, Software engineering. Software engineering is the branch of engineering that deals with the production of software according to engineering principles. The software process begins with the Requirement engineering phase, requirement engineering is a systematic process of developing requirements through an iterative cooperative process of analyzing the problem, documenting resulting observations in a variety of representation formats and checking the accuracy obtained[2]. The success or failure of any software product is highly dependent on the requirement engineering phase assuming the planning stage has been completely addressed. Industry statistics point to poor Software requirements

document as the root cause in about 50% percent of unsuccessful project [3]. It is generally accepted that the earlier the risk are identified the better as it is more expensive to fix at a later stage in the software development [3]. The Software requirements document is the first tangible representation of the system to be produced. It is designed to foster communication between the technical stakeholders such as analysts, developers, testers and customers. The Software requirements document contains both functional requirements and Nonfunctional requirements stating the functions which the system should perform and the constraints under which it should operate. If the quality of the Software requirement document is poor the project is at risk before it begins [4].

Hence, it has become imperative to establish standards to facilitate the specification and improvement of quality requirements which can also be referred to as Software requirements quality assurance. In [5] the authors synthesized the different quantitative indicators of quality desired properties of the requirements, these properties includes testability, consistency, completeness, understandability, unambiguity, traceability etc. However, among other quality factors of software requirements, testability metrics of software requirement has an effect on the resulting software to be developed because it means the requirements cannot be verified at the end of the project for pass or fail metrics. Manual requirement verification process has been introduced to identify defects in Software requirements [6], but this process poses some problems among which are: it requires linguistic and technical knowledge of the reviewer, time consuming and error prone in the case of large number of software requirements. Hence, in this work, a natural language processing based model would be presented to assure the quality of software requirements based on testability metrics and also address the limitation of the manual requirement review process. The Section 2 of this paper addresses the theoretical background, Section 3 gives a description of the methodology used for achieving the NLP based model, Section 4 presents the results of our experiments as well as the findings, Section 5 contains the conclusion, and Section 6 contains the future research directions.

2 LITERATURE REVIEW

This section provides the theoretical background related to the work, this project has its background in Software engineering, Artificial Intelligence and Natural language processing. Hence, they will be explained in this section.

2.1 Software Engineering

Software Engineering can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software [7]. It can also be defined as the application of engineering methods to the development of software; it describes not just the production of software but also the associated documentations, models, methods and theories to support software production. Software development process involves the following stages which are represented in figure 1

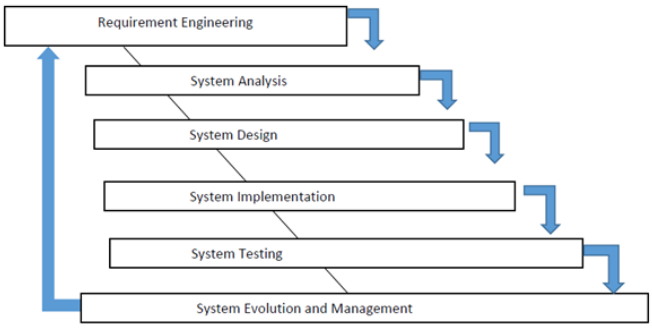

Figure 1 System development life cycle[7]

From figure 1 Requirement engineering starts the development process of a software product. This stage involves obtaining and eliciting set of requirements for a system to be developed. It involves the process of establishing what services are required, and the constraints on the system's operation and development. The stages involved in Requirement engineering includes Requirements elicitation, Requirement analysis, Requirements specifications, Requirements verifications and Requirements management [8]. Requirements engineering is the activity that transforms the needs and wishes of customers and potential users of computerized systems, usually incomplete, and expressed in informal terms, into complete, precise, and consistent specifications [9].This activity is arguably the most crucial activity in system development, because errors made in the early requirements specification phases are the most costly to repair once the system has been implemented. [10]. It is also a most delicate one, as it requires heavy involvement of requirements stakeholders, and a consensus between them, as well as between requirements stakeholders and system designers, who have quite different backgrounds and concerns. Figure 2 shows the Requirement engineering process.

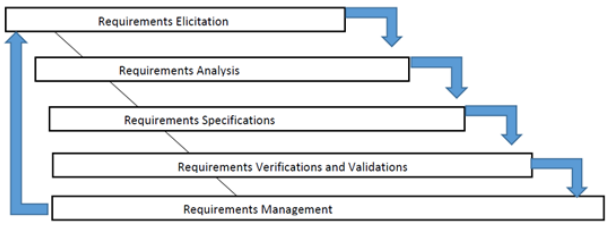

Figure 2 Stages in Requirement engineering

From figure 2 the third stage of requirement engineering process is the Requirements verification and validation phase. This phase involves the process of checking requirements to ensure they define what the customer really wants. It also involves finding problems with the requirements. Requirements validation is important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system has been developed. Amongst other checks like validity, consistency, completeness, ambiguity etc., Testability checks has an overall impact on the resulting software to be developed. To reduce the potential for dispute between customer and contractor, system requirements should always be written in a way that they can be tested [11].

As presented in Section 2.1 the Software requirements document represents the input to be delivered to developers of a system, it is therefore essential that this document is free from defects. Hence, this work aims to develop an NLP based model to assure the quality of Software requirements document.In order to do this, various techniques would be used. Section 2.2 gives an overview of these techniques, which includes Natural language processing (NLP) and Information extraction techniques.

2.2 Artificial Intelligence

Artificial Intelligence (AI) is a branch of computer science that is concerned with making a computer, or a software think intelligently like humans. AI is accomplished by studying how the human brain thinks and how humans learn, decide, and work while trying to solve a problem. It involves using the outcomes of this study as a basis for developing intelligent software and systems [12]. Amongst other dominant fields of AI such as Gaming, vision systems, Expert systems, speech recognition, handwriting recognition, intelligent robots. Natural language processing is majorly concerned with this work and it is explained in section 2.3

2.3 Natural Language Processing

Natural languages are those languages spoken or written by humans for the purposes of communication. Natural Language processing (NLP) can be defined as a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels, for the purpose of achieving human-like language processing of tasks and applications. NLP can also be defined as the branch of computer science focused on developing systems that allow computers to communicate with people using everyday language. [18]

The field of NLP involves making computers to perform useful tasks with the natural languages for humans use. The input and output of an NLP system can be either Speech or Written Text.

Another major technique to be used in this project is Information extraction technique and this is further explained in section 2.4

2.4 Information Extraction (IE)

This is a technique used for extracting structured information from an unstructured text. Information extraction is used to identify instances of a particular pre specified class of entities, relationships and events in natural language texts. It also involves the extraction of the relevant properties (arguments) of the identified entities, relationships or events [17]. The task of Information Extraction (IE) is to identify a predefined set of concepts in a specific domain, ignoring other irrelevant information. This technique would extract the main requirement phrases from SR documents. These extracted phrases would be labelled using semantic role labelling. The labels are the thematic roles or relations that were introduced in the generative grammar according to Guber [18].

Several works have been done to assess and discover defects in Software requirements document. We will review a few noteworthy studies by categorizing them according to their approaches in the following Sub-sections.

2.5 Review of Related Works

In [13], a tool was proposed called QUARS (Quality Analyzer for Requirements Specification). QUARS syntactically parses the sentences using the MINIPAR parser, then it combines both lexical (part-of-speech tags) and syntactical information to detect specific ambiguity indicators of poor-quality requirements specification. In their paper, however, the quality indicators seem to be mostly based on specific keywords, rather than on more general classes of words. At every stage of processing, QUARS requires the use of a different modifiable dictionary, which seems to be manually created and modified for a particular stage of processing and for a specific problem domain by the requirements engineer. Their idea seems to be dependent on using these special dictionaries, the relevance and practical usefulness of which are uncertain. Again, their quality measurement metrics were not well defined to characterize a text as ambiguous.

In [15], a prototype tool was developed to automate the analysis of requirements specifications documents to determine whether or not they satisfy some quality metrics. Their method was based on combination of Natural language processing techniques and specialized dictionaries, their tool performed lexical and syntactic analysis to identify keywords, phrases, or adverbs that reveals specific types of weakness present in requirements, experiments performed shows that the tool efficiently identified these defects with an error rate of less than 5%,

however their tool didn't take into consideration semantic analysis.

In [14], a requirement analysis method was proposed based on domain ontologies, where there is a matching between software requirements specifications and the domain ontology that represents the semantic components. Inference rules were applied to the ontological elements which was able to semantically process the requirements document. Their method was efficient to identify inconsistencies, incompleteness in requirements specifications. However it was domain specific.

In [16], a method was proposed for analyzing requirement specifications document using machine learning, their method was based on standard metrics that represents the characteristics that an expert takes into consideration, when deciding on the good or bad qualities of requirements.They did this using a classifier, a classifier was trained with the already classified requirements by human experts, based on the training, their method could classify a requirement as bad or good, their method was efficient to emulate the quality metrics of the expert with an average of 86.1 accuracy. However training based on human quality metrics is not too efficient as humans also are prone to making decision errors.

3 RESEARCH METHODOLOGY

This section provides the approach for the development of the NLP based model and Software Requirement Quality Assurance System (SRQAS). This involves text preprocessing, text chunking, thematic roles extraction, and Non-testability application. Software requirement were gotten from Promise data repository [19], these requirements would be preprocessed by sentence splitting, tokenization, POS tagging, to enable the efficient extraction of phrases and subsequent text analysis phase. In order to detect non-testable requirements, these sentences need to be chunked into verb phrases, noun phrases, and adjectival phrases. After these requirements have been chunked, a thematic role extraction module would be developed to create patterns and automatically extract all the thematic roles in requirements sentence in order to identify action, modality, action, theme, goal and fit-criteria.

After all thematic roles have been identified a non-testability module would be developed to automatically extract requirements without fit-criteria and identify passive voice defects to flag non-testable requirements.

The next section goes into the detailed approach for the development of the NLP based model that is discussed in this journal.

4. Development of NLP BASED MODEL for SRQAS

The primary stage of the development of this model is the text preprocessing stage. Software requirements are processed by only extracting requirement sentences from SR documents. The extracted requirement sentences are further preprocessed by tokenizing, sentence splitting, and performing morphological analysis. At the end of this phase we would have preprocessed sets of requirements.

The next stage involves syntax analysis stage; here the requirement sentences are parsed to identify the structure and dependencies between words. Part of speech tagging would be used to identify various part of speech tags such as noun, verb, adjective etc. This must be done to prepare the ground for the next extraction phase.

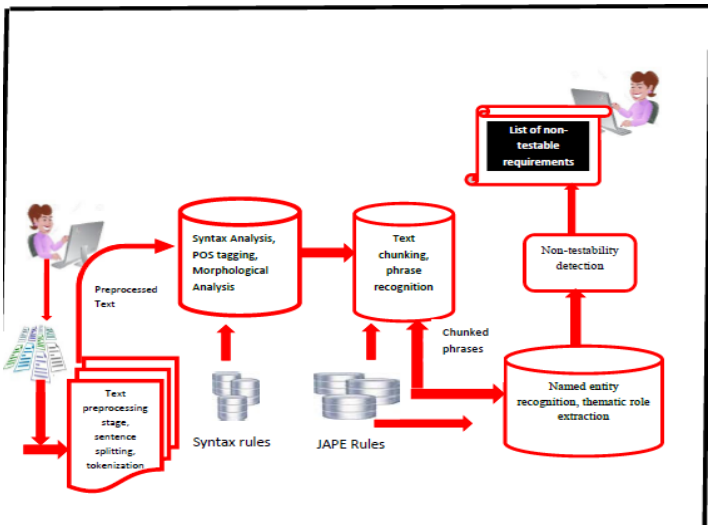

Figure 3 System Architecture of SRQAS

After part of speech tags has been assigned the next stage is the phrase recognition and text chunking stage, it consists of identifying segments or chunk of words that are syntactically related in a given requirement sentence. Thematic roles recognition involves the assignment of thematic roles to the syntactic constituents of a sentence; the relation between two or more entities is detected and assigned a thematic role using grammar patterns and rules. After all possible thematic roles have been identified such as agent, modality, action, theme, instrument, goal and fit-criteria. Our developed application would search through labeled thematic roles and flag non-testable requirement based on the existence of fit-criteria. Also action role in a sentence would be analyzed for passive voice defects.

4 Discussion of Research Findings and Results

Testing is the process of evaluating a system or its components with the intent to certify the model meets the purpose for which it was built. Information extraction technique was used for this system; the task of Information extraction is to identify a predefined set of concepts in a specific domain.Performanceevaluation of this work was based on precision, recall and F-measure. Using a Software requirements data size of 100, an experiment was performed by running the requirements through SRQAS. Fig 4 shows the screenshot of the non-testable requirements identified by the developed SRQAS. Figure 5 shows the values gotten from the performance evaluation of the system. It was observed that the precision rates was 85%. The software requirements document verifications was done manually to identify requirements that were not testable.It was observed that the precision rates of the manual process was 79%.

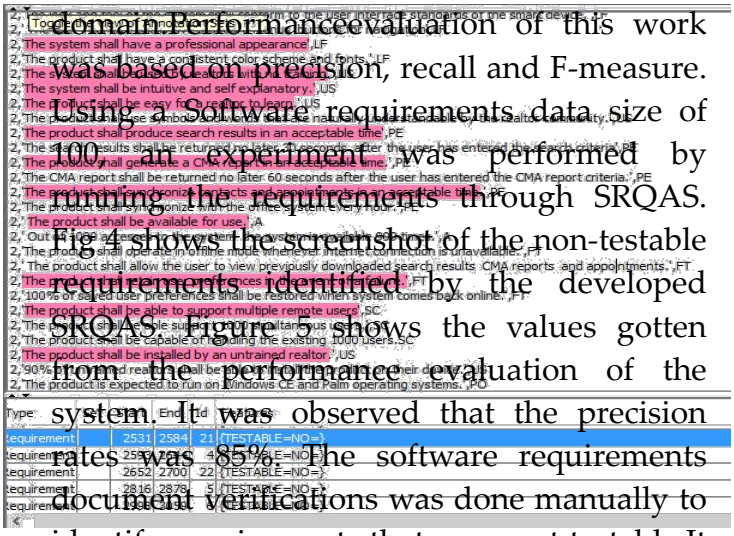

Figure 4 Screenshot of Non-testable requirements

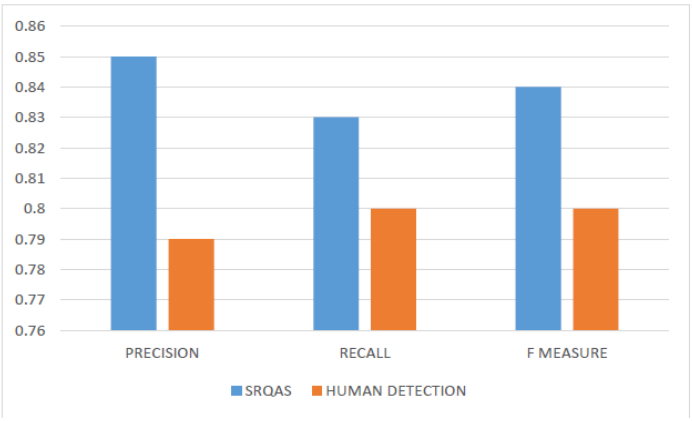

Figure 5 Graphical Result of SRQAS and Manual Requirements validation methods.

5 Conclusion

This paper presented a developed natural language processing based model for quality assurance of software requirements whichaddresses the problem of poor software quality attributed to defects in software requirements document. The contribution to this research area is the developed NLP based model to assure the quality of software requirements based on testability metrics. Evaluation of this model was done using information extraction performance metrics of precision, recall and f-measure. The developed model and system outperforms the human assessment methods with 85% precision rates as compared to 79% precision rates.

NLP support for quality assurance of software requirements has significantly increased over the years. Further improvement can be made on this work by extending the model to address other software requirements quality metrics like ambiguity, consistency, completeness and understandability metrics

.

References

[1] S.L. Pfleeger, and M. J. Atlee. *"Software Engineering theory and Practice". Pearson Higher Education, 2009 New Jersey, U.S.A.*

[2] K. Verma, and A. Kass. "Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents". *In Proceedings of the ISWC, LNCS 5318,* (2008). *pp. 751-763*

[3] A.V. Lamsweerde. "Requirements Engineering in the Year 00: A Research Perspective". *In Proceedings of the 22nd International Conference of Software Engineering*2009, *U.S.A.*

[4] E., Kamsties, D., Berry, and B. Paech. "Detecting Ambiguities in Requirements Documents using Inspections". *In Proceedings of the First Workshop on Inspection in Software Engineering, 2008 Germany.*

[5] ISO/IEC 25010:2011 -Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models, Technical report. (Accessed August 2016)

[6] B.W. Boehm. *Software Engineering Economics. Prentice- Hall, Englewood Cliffs, New Jersey, U.S.A, 2005, pp. 563-575.*

[7] I. Sommerville, (2001). *Software Engineering 8th edition, International Computer Science, Addition – Wesley Longman Publishing co.inc, Boston MA USA.*

[8] W.M, Wilson, L.H, Rosenberg, and L.E. Hyatt. "Automated Analysis of Requirement Specifications". *In Proceedings of the Nineteenth International Conference on Software Engineering (ICSE-97), 1997 Boston, U.S.A.*

[9] J. Mund,"Quality Assessment of Requirements Specifications using Metrics". Retrieved from http: www.jot.fm, (Accessed November 2016).

[10] Cheng, B.H and Atlee, A.M. (2007). Research Directions in Requirements Engineering. *In Proceedings of the future of Software Engineering FOSE. pp. 285–303*. (Accessed May 2016).

[11] F.Fabbrini,, M.Fusani,, S.Gnesi,"An Automatic Quality Evaluation for Natural Language Requirements". *In Proceedings of the 7th International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'01,*

Interlaken, Switzerland.

[12]J. McCarthy, (2005). *Circumscription: a form of non-monotonic reasoning. Artificial Intelligence.*

[13]Parra, B., Dimou, C., Llorens, F. and Moreno, M. (2015). "A Methodology for the quality assessments of Natural Language Requirements using Machine learning". *In Proceedings of the International Conference on Requirement Engineering.*

[14] H.Kaiya, and H. Saeki. "Requirements Analysis Lightweight Semantic Processing Approach". *In Proceedings of the 5th International Conference on Quality Software.2010, Canada.*

[15] A. Rojas, and M. Barrantes,. "Automatic Detection of Language Issues affecting Accuracy, Ambiguity, and Verifiability in Software Requirements written in Natural Languages". *In Proceedings of the 5th Malaysian Conference on Software Engineering. 2010 Malaysia*

[16] J. Tang, and M. Hong."Information Extraction Methodologies and Applications". *In Proceedings of the 5th International Conference of Semantic Web* (*ISWC*), *Asia, 2004* pp.640-653.

[17] D.Jurafsky, and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 2004, 1st ed. Prentice Hall. U.S.A*

[18] G.Anderman, and M. Rogers. *In and Out of English: For Better, For Worse? Multilingual Matters Ltd., 2005, United Kingdom*

[19] Promise Repository Empirical Software Engineering Data. Retrieved from PROMISE Software Engineering Repository. https://code.google.com/p/promisedata/wiki/nfr. North Carolina State University Department of Computer Science. (Accessed November, 2016).