Automated Natural Language Requirements Analysis using General Architecture for Text Engineering (GATE) Framework

Ahmad Mustafa, Wan M. N. Wan Kadir and Noraini Ibrahim
Department of Software Engineering, Faculty of Computing,
Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.
mahmad8@live.utm.my

***Abstract*—Stakeholders exchange ideas and describe requirements of the system in natural language at the early stage of software development. These software requirements tend to be unclear, incomplete and inconsistent. However, better quality and low cost of system development are grounded on clear, complete and consistent requirements statements. Requirements boilerplate is an effective way to minimise the ambiguity from the natural language requirements. But manual conformance of natural language requirements with boilerplate is time consuming and difficult task. This paper aims to automate requirements analysis phase using language processing tool. We propose a natural language requirement analysis model. We also present an open source General Architecture for Text Engineering (GATE) framework for automatically checking of natural language requirements against boilerplates for conformance. The evaluation of proposed approach shows that GATE framework is only capable of detecting ambiguity in natural language requirements. We also present the rules to minimise ambiguity, incompleteness, and inconsistency.**

***Index Terms*—General Architecture for Text Engineering; Natural Language Processing; Requirements Engineering; Requirements Incompleteness.**

I. Introduction

Requirements engineering phase is an important and effortful task during software project development. Often the software requirements specifications are stated in natural language (NL) and NL requirements may be adorned with ambiguity and incompleteness [1].

Software Requirements Specification (SRS) is used to document software requirements. It contains both functional and non-functional requirements. SRS document provides the basis for all subsequent project planning, design, coding, and testing [2].

Requirements specification analysis is considered as a challenging task because it is responsible for transforming the real-world problems into verifiable computer models [3]. According to Young et al. [4], 85% of the software errors are due to requirements defects. If defects are not removed at the early stage of the development, the cost of fixing these errors would intensely increase in the subsequent phases of development.

Standish group [5] highlight the three main reasons for project failure namely: less user involvement, lack of executive management support, and unclear or ambiguous requirements. There are other factors of project failures, but with these three elements chance of failure increases dramatically.

Normally requirements are specified in a formal (or semi-formal) way to remove the ambiguity from the requirements specifications. The UML, Finite State Machine and other abstract methods are used to formally specify the requirements [6]. However, formal specification methods are not widely accepted for several reasons. For examples, a formal model is difficult to use for communication with non-technical personnel. Additionally, extensive training is required for the implementation of formal models which is a time-consuming and expensive process.

Likewise, another way to address the ambiguity is the use of requirements template or boilerplate. The boilerplate is a blueprint for the syntactic structure of individual requirements [7]. Figure 1 shows the stages of Rupp's requirements template/boilerplate proposed by Pohl [7]. It is used to minimise the ambiguity effect in natural language. It consists of six syntactic stages: First, optional condition i.e. legal obligation for requirements are determined. Second, the core of each requirement is specified i.e. print, save or calculate. Third, the functional requirements activities are classified into three relevant types: (i) requirements that performs the process autonomously, (ii) requirements that provides the process as a service for the user, and (iii) interface requirements. Fourth, the missing processes in the previous stage are completed. Fifth, the potentially missing objects and adverbs are identified and added to the requirements. Finally, during the last stage, logical and temporal conditions are identified.

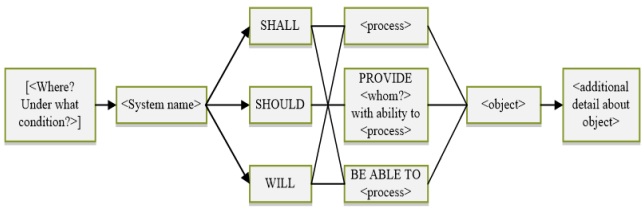

Figure 1: Rupp's Requirements template [7].

In this paper, we propose a natural language requirement analysis model which may be used to resolve the issues of natural language requirements. We also presented General Architecture for Test Engineering (GATE) framework using Natural language processing techniques. The tool automatically conforms the correct use of boilerplate in natural language requirements statements. Thus, annotated requirements will become unambiguous, complete and

consistent.

This paper is divided into the following sections: Section I explains the role of requirements specification analysis in requirements engineering, challenges during requirements specification analysis and use of boilerplate to diagnose ambiguity. It is followed by section II that discusses related studies in requirements specification analysis using NLP, and provides an overview of General Architecture for Text Engineering (GATE) Framework. Section III shows approach proposed model for natural language requirements analysis using GATE framework. Section IV explains the evaluation results of natural language requirements of the coffee vending machine. Section V is a summarization of discussion. Finally, the conclusion of work and future work is in Section VI.

II. RELATED WORKS

Related work is categorized into three sub-sections: Section A summarizes the studies that use NLP techniques in natural language requirements. Section B elaborates NLP analysis that can be used during requirements specification analysis and Section C is an overview of information extraction open source GATE framework.

A. NLP in Requirements Engineering

NLP is a promising approach which can be applied to process natural language requirements. It makes easy to manage the requirements of a complex system. Numerous studies are conducted on natural language processing techniques in software requirements analysis.

Mala and Uma [8], proposed a method of extraction from natural language (NL) requirements and transforms it into object-oriented elements of the system. Firstly, part of speech tags are assigned to each word of NL requirements. Secondly, to resolve the ambiguity issue natural language text is normalised. To map the natural language requirements into object-oriented modelling language following rules are implemented. Such as noun in natural language are converted into classes, methods in classes from verbs and attributes from adjectives. Accordingly, user requirements are mapped using part of speech tagging into object oriented programming language elements.

Likewise, another approach [10] which implement automatic mining of class diagram from NL requirements. NLP techniques are used for extraction and nouns converted into classes and verbs into relations. The authors developed a tool in Visual Basic.Net with the name of "Automatic Builder of Class Diagram (ABCD)". The tool integrates OpenNLP library and machine learning based toolkit. ABCD tool analyses the text and extracts all required information to create the corresponding XMI document. The tool generates XMI document exported to open-source ArgoUML tool to view the resulting class diagram [10].

Similarly, Almeida Ferreira and da Silva [9], presented a new socio-technical approach to reducing Requirements ambiguity and inconsistency. Requirements engineering process is aligned with Model-driven Engineering paradigm. The authors proposed a tool for automatic extraction of natural language requirements and verification of requirements models. The authors used a Wiki-based tool as an example for validation of approach.

Furthermore, Masuda et al. [12] presented a method to discover ambiguous requirements at an early phase of development. System test cases are generated from natural language requirements by focusing on UML Testing Profile (UTP) behaviour. The authors suggested three levels of rules:

i. The class is generated from noun of the sentence. The action is generated from verb and attributes are generated from the adjective.
ii. The verb is a message between two classes.
iii. Order of sequence of objects is the order of description in the requirements

Fatwanto [1], presented an approach for natural language requirements analysis using Reed-Kellogg English sentence diagramming system. It transforms natural language requirements into a scenario table.

Another study presented an automated approach in which manual and projects reports are used as input of the system. The approach has further following steps: (i) Input of manual documents in NLP based system. (ii) NLP techniques are applied to extract the scientific requirements. (iii) Specific patterns rules are defined in NLP and matched with extract requirements. Authors applied the approach study in three different scientific domains specifically in Seismology, building performance and Computational Fluid Dynamics. According to authors, 78-97% of requirements are correctly extracted using their proposed approach [11].

B. Types of Requirements Analysis Methods using NLP

The following NLP analysis methods are applied during requirement analysis phase [13]:

i. Lexical and syntactic analysis to identify the vague, incomplete and inconsistent requirements.
ii. Statistical and semantic techniques to identify the similar or duplicate requirements, and to detect the interdependencies among the requirements.
iii. Lexical and semantic analysis methods in order to identify the non-functional requirements and to classify them according to functionality (Functional / Non-functional) and for non-functional ones to sub-functionality (security, usability, adaptability etc.).

C. GATE Framework

General Architecture for Text Engineering (GATE) is a Java collection of tools initially developed at the University of Sheffield beginning in 1995. GATE framework is mostly used worldwide by companies, a community of scientists, teachers, and students for natural language processing tasks and information extraction from natural language in different languages [14].

GATE framework is software architecture for Language Engineering (LE). LE is defined as "a discipline or act of engineering software systems that perform tasks involving processing human language. Both the construction process and its outputs are measurable and predictable. The literature of the field relates to both applications of relevant scientific results and a body of practice [15]".

GATE components come in three flavours [15], as depicted in Figure 2:

i. Language Resources (LRs) consist of lexicons, corpora or ontologies resources;
ii. Processing Resources (PRs) consist of basic algorithmic, parsers, generators or ngram modellers;
iii. Visual Resources (VRs) consist of visualisation and editing of GUIs components.

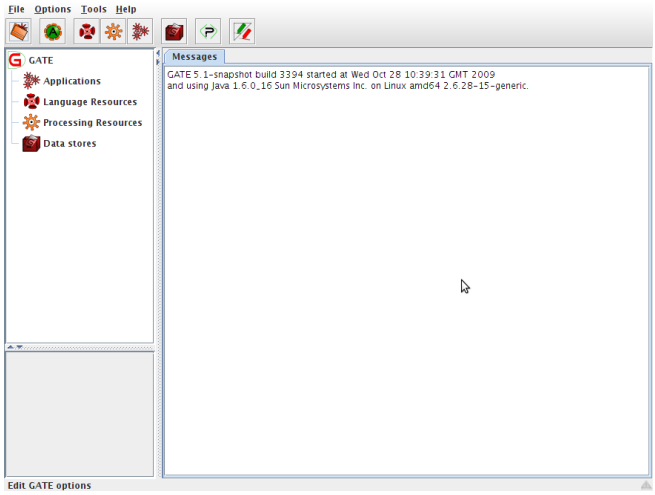

Figure 2: Main Window of GATE Developer [15]

D. Java Annotation Patterns Engine (JAPE)

JAPE is a patterns engine which recognises regular expressions in annotations of documents. JAPE grammar has two set of phases namely left-hand side (LHS) and right-hand side (RHS). Both phases consist of a set of patterns/actions rules. The LHS of the rules consists of an annotated pattern description. The RHS consists of annotated manipulated statements. The LHS is the prior part of sign '-->' and subsequent part is RHS [15, 16]. Figure 3 shows the JAPE rule example.

```
Phase: Jobtitle
Input: Lookup
Options: control = appelt debug = true

Rule: Jobtitle1
(
 {Lookup.majorType == jobtitle}
 (
  {Lookup.majorType == jobtitle}
 )?
)
:jobtitle
-->
 : jobtitle.JobTitle = {rule = "JobTitle1"}
```

Figure 3: JAPE rule example

III. PROPOSED MODEL

The proposed model as shown in Figure 4 consists of the steps to refine the natural language requirements during software requirement analysis in order to resolve the issues of ambiguity, incompleteness, and inconsistency. In this proposed model, there are three main steps that should be followed i.e. Text extraction, Boilerplates checking and NL requirements quality checking.

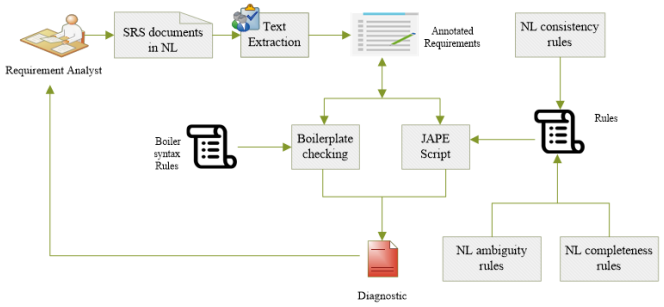

Figure 4: Natural Language Requirement Analysis Model

A. Text Extraction

In the first step, as shown in Figure 5, NL requirements document is segmented into sentences and each sentence is further subdivided into words and numbers. This procedure is called tokenization. Next, each sentence is tagged with parts of speech like verb, nouns, adjectives, and proposition. This procedure is named as part of speech (POS) tagging and it acts as bases for next step. It is known as Name Entity Recognition (NER). This step categorises the sentences in names of persons, organisations, locations, expressions of times, quantities, monetary values, and percentages. The last step is actual extraction Noun Phrase (NN) and Verb Phase (VP) are handled by a separate module.

The input requirements after going through NLP process are marked with tokens, sentences, part of speech tagging, named objects, NN and VPs. These annotated natural language requirements used for automatic conformance against boilerplates and to diagnose ambiguity from natural language requirements.

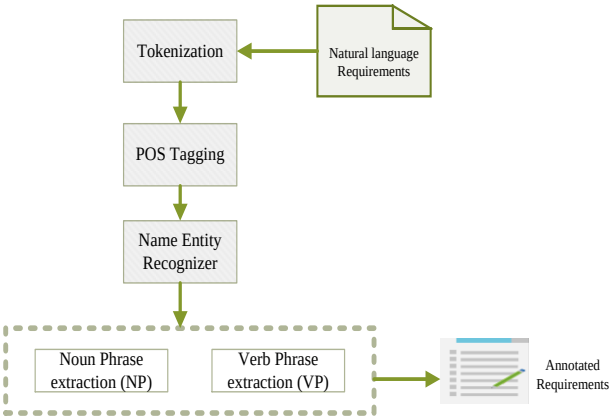

Figure 5: NLP Pipeline for Text Extraction System

B. Boilerplate checking

In this step, the annotated requirements document produced by the text extraction in the previous step is used as an input for Boilerplate checking. Boilerplate checking rules are defined in BNF grammar. The main purpose of these rules is to tag each requirement statement areas either it conforms to the boilerplate rules or not. The Rupp's requirement boilerplate rules, as shown in Figure 1, are used during conformance checking. If necessary, any other boilerplate also can be used for this purpose.

Figure 6 shows BNF grammar which contains characteristic of Rupp's requirement boilerplate. This gets the input of annotated requirement produced by text mining process.

```
<Template-CAB>::=
<opt-condition> <NP> <VP-starting-MD> <NP>
<opt-detail> |
<opt-Condition> <NP> <MD> "PROVIDE" <NP> "WITH THE
ABILITY" <infinitive-VP> <NP> <opt-Details> | <opt-
Condition> <NP> <modal> "BE ABLE" <infinitive-VP> <NP>
<opt-Details>
<opt-Condition>:: = "" |
<conditional-keyword> <non-punctuation-Token>* "," <opt-
Details>:: = "" |
<Token-sequence-without-subordinate-conjunctions>
<MD>:: = "SHALL" | "SHOULD" | "WOULD" <conditional-
keyword>:: = "IF" | "AFTER" | "AS SOON AS" | "AS LONG
AS"
```

Figure 6: BNF grammar for Rupp's boilerplate

These BNF grammar rules are used to communicate with JAPE. In Figure 7, JAPE script is communicating with BNF grammar and automatically validating each sentence against autonomous, user interaction and interface requirement.

```
1.Rule: MatchCabRequirementFragment
2.(
3. (({Condition}{NP}) |
4. ({NP}))
5. ({VP, VP.startsWithValidMD == "true",
6.! VP contains {Token.string == "provide"}})
7. ({NP})
8.): label -->: label.Conformant_Fragment =
9. {explanation = "Matched pattern: CAB"}
```

Figure 7: JAPE script for conformance of Rupp's boilerplate

C. Natural language quality checking

In this step, the natural language requirements are further verified that requirements statements are complete, unambiguous and consistent. Therefore, the aim of this step of this step is to diagnose incomplete, ambiguous and inconsistent requirements. Next is a detail of the different type of ambiguity, inconsistency and incompleteness.

1) Ambiguity Issue

A requirement is considered as ambiguous if it has more than one interpretation [17]. Ambiguity in natural language requirements may come under following types [18]:

a) Analytical Ambiguity:

Analytical ambiguity occurs when some part of the sentence can have more than one role within a sentence.

b) Attachment Ambiguity:

Attachment ambiguity occurs when some part of the sentence can be attached to more than one other part of the sentence.

c) Coordination Ambiguity:

Coordination ambiguity occurs when more than one conjunction (e.g. and or) is used in a sentence or when the conjunction is used with a modifier.

d) Scope Ambiguity:

Scope ambiguity occurs when using quantifier operators such as all, some, etc. and negation operators such as no, not, etc.

e) Referential Ambiguity:

Referential ambiguity happens when a word or phrase is referring to two or more properties or things. It has further categories:

i. Antecedent: A reference word or phrase.
ii. Anaphora: Refers to a previous expression in the same or in a previous sentence.

2) Incompleteness Issue

A requirement is incomplete if necessary information is missing for implementation or no enough information for the design process to continue further. This incompleteness of requirement statement will cause of incomplete software specification [17].

3) Inconsistency Issue

An SRS document is inconsistent if there are conflicts between requirements or terms are used in different ways in different places. The dependency analysis is performed to diagnose the inconsistency in natural language requirements. We extract subject, verb, and action from natural language requirements with text extraction process. We used the NLP parser and conducted dependency analysis on natural language requirement. We have the focused on fine grains of dependency of one word with another word within requirements statements. When the two words are connected by a dependency relation, one of them is head and other is dependent. The dependency parse represents the syntactic structure of a sentence in terms of binary relations between tokens.

Requirements statements can be presented according to following structure:

Requirements ⟵ Subject | Verb | Action

A requirement statement an activity or action performed by the user who may affect or changes the state of the object. Subject signifies the user who executes the behaviour of the verb and verb defines as any activity which is taken by the subject or user.

IV. Evaluation

Natural language requirements of the coffee vending machine were used as an example for evaluation of approach. Firstly, the requirements were manually inspected by the analyst. In next phase, the requirements were analysed using pipelining. The accuracy of approach is measured using statistic metrics precision, recall, and f-measure. Using annotation diff tool in GATE framework precision, recall and f-measure were checked. Precision and recall were at 88.5% and 93.8%. Similarly, harmonic means of precision and recall was 92.5%.

Our evaluation suggestion that GATE framework is effective for only ambiguity detection. On the other hand, the framework is not capable of detecting inconsistency and dependency analysis in natural requirements. In order to produce the annotated natural language requirements according to our model mentioned in Section III, we discuss steps, rules, and execution process. We implement Stanford NLP parser for parsing natural language requirements. For dependency analysis, we also implement Stanford NLP parser.

V. Discussion

In this paper, authors discussed natural language software requirements analysis and different related studies that applied NLP techniques on natural language requirements [1, 8-10, 12]. A requirement analysis model is proposed to address the ambiguity, inconsistency, and incompleteness. It is our belief that the NLP knowledge domain can be used in software requirements analysis phase through which software having better quality are developed. Currently, many researchers are doing the research on applications of NLP in the areas like Text summarization, Auto-completion, Part-of-speech tagging Sentiment analysis, and Optical Character Recognition (OCR).

In this research, we proposed an NL requirement analysis model for requirement analysis. Moreover, we also presented GATE framework for automation is analysis process. This

study does not have detail experimentation on issues of natural language, it only presented a holistic model for requirements analysis. Manually analysing complex natural language is a tedious and time-consuming task. With the help of open source, GATE framework analyst can easily analyse the issues in natural language requirements.

Software natural language requirements analysis is the key factor for the achievement of a requirement engineering phase in the software development process. Whereas the success of requirement analysis phase is dependent on the user involvements and clear statements of requirement.

NL Requirements should be analysed to provide a degree of quality in a requirement set. For automated analysis of NL requirement specification, there is suggested a model as shown in Figure 4. These procedures must be followed in requirements engineering phase. If we can recognise the maximum fault and area of concerns in the early phase of requirements. It will decrease the possible number of errors in the system, loss of cost and time as well.

VI. CONCLUSION

This paper focuses on natural language requirements analysis using NLP technique. The study introduced an approach to automated NL Requirements analysis process using General Architecture for Text Engineering (GATE) framework. To diagnose the ambiguity, an automated method is proposed for conformance of natural language requirements against Rupp's boilerplate. The study suggests that ambiguity rules can be applied using GATE framework. Similarly, inconsistency checking using dependency analysis and NLP Stanford parser. Furthermore, the study also proposed a model for refinement and automation in the natural requirements analysis. In future, we have the plan to conduct separate comprehensive research studies on each topic specifically on ambiguity, inconsistency, and incompleteness problems of natural language requirements and automation in suggestion process mentioned in Figure 3. We will demonstrate the benefits of GATE framework in an industrial context and use it as research tool in diagnosing the ambiguity from NL requirements. Another direction of our future work is to study the effect of automated conformance of boilerplates on software development methodology in term time, effort, and precision of output product.

ACKNOWLEDGEMENT

The authors would like to express their deepest gratitude to Research Management Center (RMC), Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education Malaysia (MOHE) for their financial support under Research University Grant Scheme (Vot number Q.J130000.2516.11H71).

REFERENCES

[1] A. Fatwanto, "Software requirements specification analysis using natural language processing technique," in *2013 International Conference on QiR (Quality in Research)*, 2013, pp. 105-110.

[2] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

[3] N. H. Ali, Z. Shukur, and S. Idris, "A design of an assessment system for UML class diagram," in *International Conference on Computational Science and its Applications (ICCSA 2007)*, 2007, pp. 539-546.

[4] R. R. Young, "Recommended requirements gathering practices," *CrossTalk*, vol. 15, pp. 9-12, 2002.

[5] S. Hastie and S. Wojewoda, "Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch," *InfoQueue*, 2015. Available at https://www.infoq.com/articles/standish-chaos-2015

[6] J. Holt, *UML for Systems Engineering: Watching the Wheels.* The Institution of Engineering and Technology, 2004.

[7] K. Pohl, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam-Foundation Level-Ireb Compliant*: Rocky Nook, Inc., 2016.

[8] G. S. A. Mala and G. V. Uma, "Automatic construction of object oriented design models [UML Diagrams] from natural language requirements specification," in *PRICAI 2006: Trends in Artificial Intelligence*, Q. Yang, and G. Webb, Eds. Berlin, Heidelberg: Springer, 2006, pp. 1155-1159.

[9] D. de Almeida Ferreira and A. R. da Silva, "A controlled natural language approach for integrating requirements and model-driven engineering," in *Fourth International Conference on Software Engineering Advances, 2009. ICSEA'09*, 2009, pp. 518-523.

[10] W. B. A. Karaa, Z. B. Azzouz, A. Singh, N. Dey, A. S. Ashour, and H. B. Ghazala, "Automatic builder of class diagram (ABCD): An application of UML generation from functional requirements," *Journal Software - Practice & Experience*, vol. 46, no. 11, pp. 1443-1458, 2016.

[11] Y. Li, E. Guzman, K. Tsiamoura, F. Schneider, and B. Bruegge, "Automated Requirements Extraction for Scientific Software," *Procedia Computer Science*, vol. 51, pp. 582-591, 2015.

[12] S. Masuda, T. Matsuodani, and K. Tsuda, "Automatic generation of UTP models from requirements in natural language," in *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2016, pp. 1-6.

[13] A. E. Yilmaz and I. B. Yilmaz, *Knowledge Engineering for Software Development Life Cycles*. IGI Global, 2011, pp. 21-33.

[14] Wikipedia, "General Architecture for Text Engineering," https://en.wikipedia.org/wiki/General_Architecture_for_Text_Engineering, ed, 2016.

[15] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, C. Ursu, and M. Dimitrov, *Developing Language Processing Components with GATE (A User Guide)*. The University of Sheffield, 2003.

[16] D. Thakker, T. Osman, and P. Lakin, *GATE JAPE Grammar Tutorial*, Nottingham Trent University 2009.

[17] B. Kiepuszewski, A. H. M. ter Hofstede, and C. J. Bussler, *Engineering Requirements with Desiree:An Empirical Evaluation*. Berlin, Heidelberg: Springer, 2000.

[18] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Technique.* Springer Publishing Company, Incorporated, 2010.