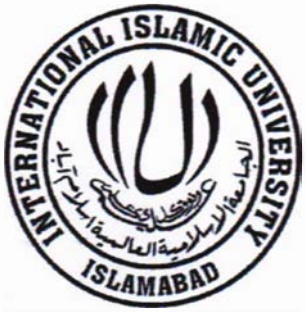
Generating Alloy Specification from Textual User Requirements written in Natural Language



By

Kiramat Rahman

161-FBAS/MSSE/F07

Supervised by

Mr. Shahbaz Ahmed Khan Ghayyur

Assistant Professor Department of CS & SE

International Islamic University Islamabad

A Thesis

Submitted in partial fulfillment of the requirements

For the award of degree of Master of Science in

Software Engineering

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
INTERNATIONAL ISLAMIC UNIVERSITY
ISLAMABAD

2012

1. Natural language processing

2. Text processing (computer science)

International Islamic University, Islamabad

Faculty of Basic & Applied Sciences, Department of Computer Science & Software Engineering

Dated: July16, 2012

FINAL APPROVAL

It is certified that we have read the thesis, entitled **"Generating Alloy Specification from Textual User Requirements Witten in Natural Language"**, submitted by KiramatRahman, Reg. No. 161-FBAS/MSSE/F07. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University Islamabad for MS Degree in Software Engineering.

PROJECT EVALUATION COMMITTEE

External Examiner:
Dr. Muhammad Afzal
Director KICSIT

Internal Examiner:
Professor Dr. Muhammad Sher
Chairman, DCS & SE
IIU ISLAMABAD

Supervisor:
Mr. Shahbaz Ahmed Khan Ghayyur
Assistant Professor, DCS & SE
IIU Islamabad

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "Generating Alloy Specification from Textual User Requirements Witten in Natural Language", in partial fulfillment of the requirements for the award of degree, Master of Science in Software Engineering at Computer Science and Software Engineering Department of International Islamic University Islamabad is an authentic record of my own work carried out under the supervision and guidance of Mr. Shahbaz Ahmad Khan. The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other University.

Kiramat Rahman
161-FBAS/MSSE/F07

I would like to dedicate this thesis to my loving parents ...

Acknowledgement

The works like this are never completed single-handed. There are many persons working behind the screen but are equally important. I take this opportunity to express our deep sense of gratitude towards all those.

First of all, humble bow before the almighty Allah for his blessings and providing the power to finish this work.

I thank the computer science and software engineering department of International Islamic University Islamabad for giving me opportunity and providing resources for carrying out the research and completing thesis work. I take the opportunity to thank my supervisor Mr. Shabaze Ahmad Khan, Assistant Professor, who has been a continuous source of inspiration for me, for his encouragement, patience and providing invaluable guidance throughout this work, especially when I was unsure about which direction to take.

From a personal side, I would like to thank my colleagues and friends, without whose support, I might not be able to finish the thesis work. My thanks are also due to Professor Dr Joo Miguel Fernandes for his valuable advice and moral support during the thesis. Last but definitely not the least, I would like to thank my parents and family members, for their support throughout the work.

Kiramat Rahman

Abstract

Software requirements are written in textual forms using a natural language. This is a very common way for writing user requirements; since it allows non-technical stakeholders to read, comment and discuss the requirements in a format that they can comprehend. However, user's requirements written in natural languages can often be ambiguous, incomplete and inconsistent, which leads to problems when developing software system. Additionally, the interpretation of requirements in natural languages can be affected by geographical, psychological and sociological factors. Thus, it is very relevant to detect and fix potential ambiguities, inconsistencies and incompleteness in the written requirements. This leads to better products, which fullfills accordance to the needs and expectations of stakeholders and reduces the development cost. We have offered a two-step approach to address these issues. In this thesis we propose techniques that generate Alloy specifications from textual user requirements written in English. This transformation is occurring in two steps. In the first step, textual requirements written in any form are rearranged according to a more some rigid format that we have proposed. Guidelines and recommendations have provided to suggest how to write the requirements according to the format. In the second step the requirements transformed into alloy specification, using the transformation rule we have defined. This step is amenable to some form of semi-automatic processing and two mode of for requirements specification has been provided.

1. A mathematical format based on the standard Alloy specification language

2. A natural language format that uses a reduced set of English words to describe the systems

These two forms, that are equivalent, should permit to easily link the Alloy specifications to the textual requirements. We can either write the Alloy specifications or check if the corresponding sentences are part of the requirements or alternatively write the sentences and obtain the Alloy specifications.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Chapter 1

Introduction

Software requirements are written, in textual forms using a natural language. This is a very common way for writing user requirements; since it allows non-technical stakeholders to read, comment and discuss the requirements in a format that they can comprehend. However, user's requirements written in natural languages can often be ambiguous, incomplete and inconsistent, which leads to problems when developing the software system. Additionally, the interpretation of requirements in natural languages can be affected by geographical, psychological and sociological factors.

Thus, it is very relevant to detect and fix potential ambiguities, inconsistencies and incompleteness in the written requirements. This leads to better products, which fullfills accordance to the needs and expectations of stakeholders and reduces the development cost.

In the literature of Software Engineering, there is no comprehensive definition of ambiguity. IEEE recommended practice for software Requirements specification, [1] says that "An SRS is unambiguous if, and only if, every Requirements stated there in has only one interpretation". The Problem with IEEE definition is that there is always someone who understands requirement specification different from someone else. Davis [9] defines ambiguity as to extract one statement of SRC and give it to 10 peoples, if they interpreted it differently than the statement is ambiguous. The problem with this definition is, what is guaranty that the 11th person will not interpret differently. Schneider et. al. [35] defines ambiguity as "An important term, phrase or sentence essential to and understanding of sys-

tem behavior has either been left undefined or defined in a way that can cause confusion and misunderstanding."

Guase et. al. [14] according to them ambiguity is either of missing information or communication error. Missing information means human make error in observation or recall and leave self evident and other fact. Communication error occurs due to general problem in writing between author and reader. Kamsties et. al. [22] defines a requirement is ambiguous if it has multiple interpretation despite the reader knowledge of the context. It does not matter either the author have introduce the ambiguity intentionally or unintentionally. He has presented comprehensive taxonomy based on this definition. They have classified ambiguities into Linguistic ambiguity (which make a requirements statement ambiguous) and software engineering ambiguities. Software engineering ambiguities are context oriented which are to be taken under consideration when requirements statements are evaluated for such kind of ambiguities. The context includes requirement document, application domain, system domain and the development domain.

Although the most recommended method to address these issues is the use of Formal requirements specification language [23] but natural language is acceptable in common way of writing requirements specification in industries. Even when formal requirement specification language is used, still the initial requirement has been written in natural language. Requirements Engineer must translate these informal requirements into formal or semiformal requirements. Initial requirement, which is written in natural language, is ambiguous and it is often not recognized during this translation. If these initial requirements are misinterpreted unconsciously, it can slip through undetected and will lead product against the need and expectation of clients. The reason is that client domain expert does not understand semiformal and formal language to detect a meaning different from their experiences or intentions. We have proposed a two step semi-automatic approach to addresses these issues. The approach presented here support both client and technical peoples by addressing the potential ambiguity, inconsistency and incompleteness of natural language through formal specification. The approach produce two modes of requirement specifications which are semantically equivalent but represent two forms that is textual form based on

reduce set of English word and mathematical form based on formal specification language alloy.

1.1 Motivation and Objectives

Requirements play a key role in the development project. It is based for design, implementation, testing and verification. The product should be developed according to the expectations and needs of stakeholder. To develop product according to the expectation of a stakeholders, initial requirements must be written in natural language. There is potential ambiguity, inconsistency and incompleteness in requirements when used informal language which is often undetected during the translation of these requirements into formal or semi-formal requirements by requirements engineer. Additionally, the client domain expert does not understand the semi-formal or formal language to detect a meaning different from their intention.

We are motivated with the fact to propose an approach for writing requirements that support both stakeholder and developer using both formal language and natural language. Our main objective is that without excluding natural or formal language despite of their weakness but to convert these weaknesses into strength. Furthermore, the weakness of natural language is that the ambiguity, inconsistency and incompleteness is addressed by formal language, and the weakness of formal language, that is complexity, is achieved by natural language. Using of informal and formal language for writing requirement specification we can achieve both preciseness and accessibility. The approach should be cost effective in terms of effort and time and lead to produce a quality product according to need and expectation of stakeholder.

1.2 Problem Statements

Initially requirements written in informal language and the requirements engineer must translate it into semi or formal language. Requirements written in informal language has a potential ambiguity, incompleteness and inconsistency and it is

often undetected during the translation of the requirements into formal or semi-formal language. The reason is the client domain expert does not understand the semi formal or formal language to detect a meaning different from their intention. These problems lead to unconscious misinterpretation of the requirements, which lead to develop a product that is not according to the need and expectation of stakeholder.

1.3 Research Question

How do we address potential ambiguities, inconsistencies and incompleteness in requirements written in natural language using alloy specification?

1.4 Thesis Organization

In Chapter 1: Introduction to the area and problem investigated in the thesis Chapter 2: considers the background information of the domain for understanding the rest of the thesis. Chapter 3: investigating the relevant literature, Chapter 4: presenting the research method to address the problem, Chapter 5: have describes detail proposed solution, Chapter 6: implementing the proposed approach with example and statistical evaluations of the proposed approach. Finally a conclusion and future work is given in Chapter 7.

1.5 Summary

The chapter presents a brief introduction of the thesis. The chapter exploring the problem addressed in the thesis and brief introduction to the area, motivation and objective of addressing the problem has been discussed and the research question derived from the problem statement. Finally, thesis organization has been given in the chapter.

Chapter 2
Background Information

Chapter 2

Background Information

This chapter introduces various domain investigated for the completion of the thesis. This chapter will support the readers in understanding the subsequent chapter. In section 2.1 we have introduced alloy and explored what is alloy and the basic structure, syntax, declaration and reserved key word of alloy specification language. Section 2.2 presents introduction to alloy analyzer, the tool that is used for the automatic analysis of requirement specification written in alloy. Section 2.3 describes requirements specification and subsequent section presents type of language used to write requirements specification. Section 2.4 narrates the basic terms ambiguity, inconsistency and incompleteness and there classification.

2.1 Alloy

In Jackson's words [20] "Alloy is an attempt to combine the best features of Z and the Object Constraint Language of UML in a lightweight notation. It takes UML's emphasis on binary relations, and the expression of constraints with sets of objects formed by 'navigations', but with Z's much simpler semantics."

Alloy is a formal specification language used to describe the structural properties, offers declarative syntax, which is compatible with graphical object model, and expressing the complex constraints with set based formula. Alloy is amenable

to fully automatic semantic analysis.

Table 2.1 shows the reserved keywords for alloy specification language. They will not be used as identifiers.

Table 2.1: Alloy Keywords

abstract	all	and	as	assert
but	check	disj	else	exactly
Extends	fact	for	fun	iden
iff	implies	in	Int	let
lone	module	no	none	not
one	open	or	pred	run
set	sig	some	sum	univ

2.2 Alloy Analyzer

There are two different kinds of tools that support formal methods, one is theorem provers and the other is model checkers. These tools are based on the characteristics of formal methods. Alloy analyzer is a slightly different kind of tool that is defined as a model finder, its work by finding models with in a limited scope as counter example to the user assertion: "Its engine takes a formula and attempts to find a model of it". Alloy which is based on first order logic (while model checking is based on temporal logic) implements techniques for finding finite models by Jackson, and Paradox by Claessen et. al. [8] are examples of a program that implements techniques for finding finite models based on first order logic, whilst model checking is based on temporal logic.

2.2.1 Alloy Analyzer Layout

Figure 2.1 shows a screen shot of the main window of the Analyzer.

2.2.2 Toolbar

Alloy analyzer main tool bar consist of new, open, reload, save, execute and show menus which provide a quick access to the most common operations Figure 2.2

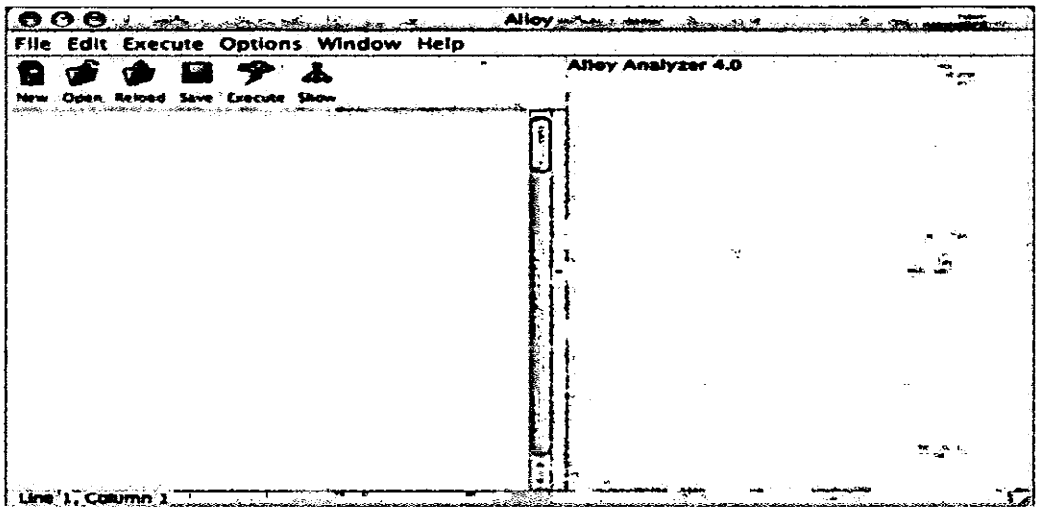


Figure 2.1: Alloy Analyzer Layout

Figure 2.2: Main Toolbar

shows a screen shot of the main window of the Analyzer.

2.2.3 Editor Panel and Message Panel

The user interface consists of the editor panel and the message panel. The relative sizes of panels may be adjusted by clicking and dragging the split bars that separate the panels.

- Editor panel: contains a tabbed text editor for modifying Alloy models. It supports, tabbing so you can edit multiple text files simultaneously. It also supports error highlighting during model compilation.

- Message panel: displays the results of analysis. Each counterexample and each satisfying instance will have a clickable hyperlink. Clicking on it will launch

the Alloy Visualizer to display the counterexample or instance. The message panel is also used for general status messages and error messages. For example,

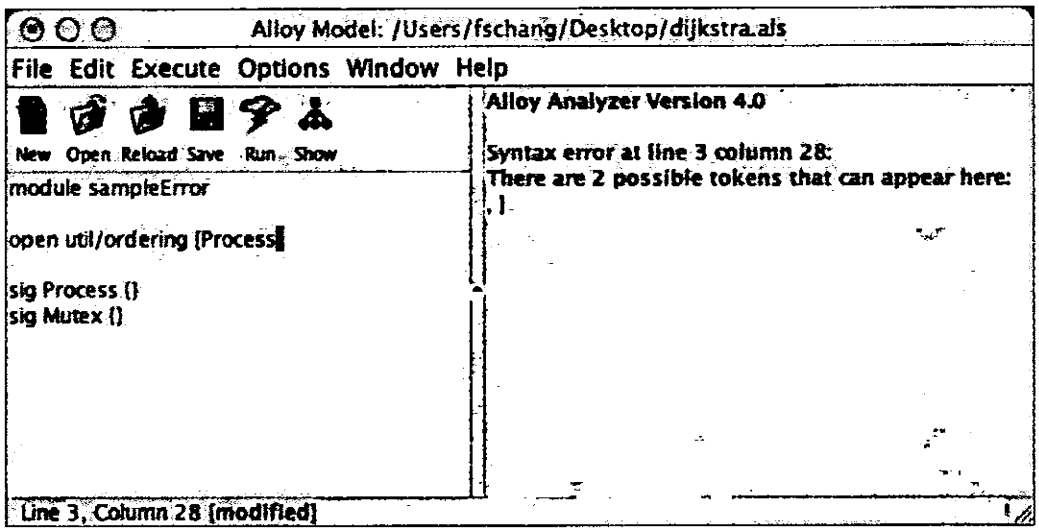


Figure 2.3: Editor Panel

if a model cannot be compiled, an error message is displayed, and the error will be highlighted in the source Alloy model as shwon in Figure 2.3

2.3 Requirements Specification

Requirement specification aims at the production of an SRS. The main purpose of a requirements document is to convey information, gathered from the customer and other sources, to the developer [11]. Specification means different things to different people; it is a written document, formal mathematical model, a collection of usage scenarios, a prototype or graphical model or combination of these [29]. To present requirement in a consistent and more understandable manner Sommerville et. al. [37] suggest a "standard template" that should be developed and used for system specification. However, flexibility is necessary to develop requirement specification, for large system, the best approach is a written document and combination of natural language description and graphical models. For small system usage scenarios are required. Requirement engineer produce the system specification, which is the final work product serve as a foundation for software engineering. It describes the function and constraint of a computer based system. System specification also describes data and controls the is in-

put to and output from the system. Requirements specification is presented by three manners. A specification can be a written document, graphical model, a formal or mathematical model, mental model or a combination of these [10; 31]. Requirement specification languages can be categorized into following classes:

2.3.1 Informal languages

Informal language is the language that the stakeholders speak in their routine life. Informal langue is natural language such as English. People can disagree on what it means. Text and recording, picture and animation fall in this category. The requirements expressed in informal language can be understood by all stakeholders, including non-technical customers. However, they are very prone to inconsistencies, contradictions, incompleteness and misunderstandings [10].

2.3.2 Semi-formal languages

Semi-formal language used some graphical notation along with a natural language to express requirements. Entity relationship diagrams (ERD), data flow diagrams (DFD) and state transition diagrams (STD) are very commonly used within industry for the semi-formal expression of requirements. They are easy to understand, and provide a good overview of the system. Such languages represent a middle way between complete informality and complete formality, and can be used for the transition from informal requirements to a formal specification [10].

2.3.3 Formal languages

Formal language provide a foundation for specification environments leading to analysis models that are more complete, consistent, and unambiguous that produce using conventional or object oriented methods. The descriptive facilities of set theory and logic notation enable a software engineer to create a clear statement of facts (requirements). The underlying concept that govern formal method are

- The data invariant, a condition true throughout the execution of the system that contains a collection data.

- The state, the stored data that a system access and alter
- The operation, an action that takes place in a system and reads or writes data to a state. An operation is associated with two conditions that is precondition and post condition.

Discrete mathematics, the notation and heuristics associated with sets and constructive specification, set operator, logic operators and sequences form the basis of formal language [30].

2.4 Natural Language Requirements and its Inherited Problems

Natural languages have associated inherited problems that are ambiguity, inconsistency and incompleteness. In this section we are exploring these problems with different perspective.

2.4.1 Ambiguity

Ambiguity is the state or quality of being ambiguous, specifically in meaning such as an ambiguous expression or word. The word, expression or requirement that is capable of being understood in two or more possible ways or senses. Requirements ambiguity is classified into following categories. There are three strategies to address the ambiguity of natural language; first strategy is learns to write less ambiguously, second is learns to detect ambiguity and the third strategy is to use a restricted natural language, which is inherently unambiguous and more precise.

2.4.2 Inconsistencies

An Inconsistency creates a situation, where some requirement may be contradicted with the other requirements. That is, we can achieve one requirement at the cost of other. Clearly it is important that inconsistencies be detected and corrected at early stages before the system's deployment

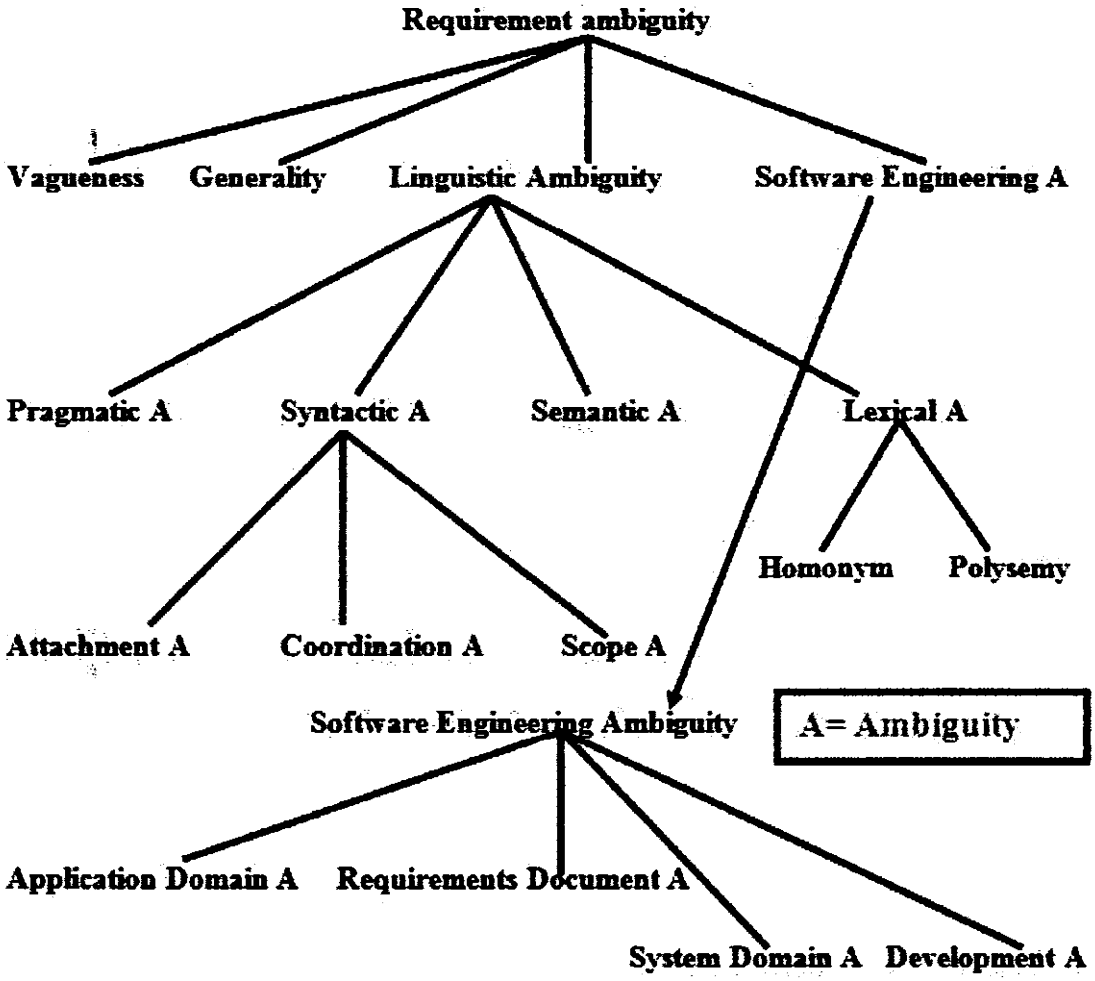


Figure 2.4: Classification of Ambiguity

2.4.3 Incompleteness

The requirement is incomplete, as a developer will not be able to implement the desired functionality without additional explanation. Technically, requirements are incomplete, not only because additional explanation is needed, but also if the developer is exercising unreasonable discretion (i.e. further explanation may be needed, but is not obtained). Of course, there needs to be a balance here. Until there is a working system, decisions are needed, and the working system can be considered to be a final statement of requirements. Therefore, the issue is not so much when a developer is exercising discretion, but where the exercise of discretion is unreasonable. For example, a database is always a representation of a set of business rules. Those rules may not have been stated up front when defining requirements, and the database developer may need to make assumptions about business rules. These assumptions should be checked, but we have seen so many instances when they have not been. 'Completeness' is also highly context dependent: a model of a system for presentation to the board that will approve the cheque with lots of zeros on the end, has a different standard of completeness from the presentation to the development engineers. In automating a set of completeness checks of a specification, you will always need to know first, what is the purpose of the statement of requirements.

2.5 Summary

This chapter presents background information that will help in understanding the rest of the thesis. A short description of the alloy specification language and alloy analyzer, the tool that is used for automatic analysis has been presented. Requirements specification definition and type of language used for writing specification has been described. The natural language problem that is ambiguity, inconsistency and incompleteness has been explored and defined.

Chapter 3
Literature Review

Chapter 3

Literature Review

In this chapter, we review different techniques, methods proposed by researchers and claims that describe need of writing requirement in Natural language and the associated inherited problems and how they have addressed these problems. Requirements communicate needs from stakeholder to developer on a development project, Ian Alexander [2], therefore requirements should be written in language that is understandable by the stakeholder. Normally, requirements are written in Natural language which is accessible and flexible Lori et. al. [36]. Requirements written in natural language have potential ambiguity, inconsistencies and are incomplete. Different tools, techniques and approaches have been presented to address these issues.

3.1 Specification in Natural Language and Ambiguity

Kamstics et. al. [23] have proposed an inspection technique to detect requirement engineering specific ambiguities in informal requirement document. In order to tailor the techniques to the context of different project they have proposed a Meta model.Chantree [6] argue that coordination ambiguities have received a little attention as compared to other syntactic ambiguity. In order to detect coordination

ambiguity they have presented heuristics and used some measure on requirement documents specification. Chantree [7] has presented a technique that provides an automatic alert to requirement engineer for the presence of dangerous ambiguity which leads to misunderstanding. They have validated there techniques to coordination ambiguity in requirement document specification.Kiyavitskaya [25] has proposed a two step approach to detect ambiguity in requirement document written in natural language. In the first step they have identify the potential ambiguity in the requirement document by applying ambiguity measuring. In the second step other tool used to identify the relevant cause of the statement that why it is ambiguous. A.Fantechi [12] has presented the use of method based on linguistic approach to detect the inherent ambiguity in functional requirement expressed in Textual (NL) use cases. They have collected metrics and Perform a qualitative analysis of the requirements. They have also investigated the use of linguistics technique for consistency check and semantic analysis. Kamsties et. al. [22] have categories ambiguities and describe the difference between linguistics and Requirement Engineering (RE) Specific ambiguities. Linguistic ambiguities are describes in RE text book and RE Specific ambiguities are context dependent. They have presented an approach to identify RE specific ambiguities on the base of Meta model. On the base of these identified types of ambiguities they have developed an improved inspection technique for natural language requirements and using check list and scenario based reading. Vincenzo et. al. [3] have build several tools and techniques that support to extract information from natural language text of the requirements and build a (semi-formal models in automatic fashion. These tools support to check and measure the consistence of these models. Alistair et. al. [28] have developed a small set of rules to address eight basic requirements problems including, complexity, omission, vagueness, ambiguity, duplication, wordiness, inappropriate Implementation and untestability. These rule set allow all natural language requirements should be written in five simple template. They have evaluated their approach through a case study.

Table 3.1 Ambiguity concept matrix

Serial number	Technique	Concept = Issues and problem addressed										
		Ambiguity										
		RE Ambiguity				Linguistic Ambiguity						
		Document ambiguity	Application Domain ambiguity	System domain	Development Domain ambiguity	Lexical Ambiguity		Syntactic ambiguity			Semantic ambiguity	Pragmatic Ambiguity
						Homonym	Polysemy	Attachment	Coordination ambiguity	Scope ambiguity		
1	[2]	X	X	X	X	Y	Y	X	X	X	X	X
2	[3]	Y	Y	Y	Y	X	X	X	X	X	X	X
3	[4]	X	X	X	X	X	X	X	Y	X	X	X
4	[5]	X	X	X	X	X	X	X	Y	X	X	X
5	[6]	Y	X	X	X	X	X	X	X	X	X	X
6	[8]	Y	Y	Y	Y	X	X	X	X	X	X	X
7	[9]	X	X	X	X	Y	X	X	X	X	X	X
8	[10]	X	X	X	X	X	X	X	X	X	X	X

Figure 3.1: Ambiguity Concept Matrix

3.2 Requirements Specification in Natural Language and Inconsistencies

Inconsistencies are one of the major problems in requirements specification. Inconsistencies mean a conflicting, contradictory description in requirements specification of the expected behavior of the system to be built or of its domain Ghezzi et. al. [16]. Vincenzo et. al. [15] describe two main sources of inconsistencies that are conflicting goals between two parties and uncoordinated change in requirements specifications. To address the issues of inconsistencies there are two schools of thought. The first schools of thought [34; 38] propose tool, techniques that inconsistencies should be corrected before further activities take place and it should be treated as an error. The second school of thought[4; 13]proposed that inconsistencies should be tolerated and can be corrected at latter stage. Vincenzo et. al. [15] argue that logic is most effectively used in number of studies for inconsistencies analysis and identification in requirements but logic is not understandable for stakeholder. Therefore they have integrated natural language parsing techniques with the default reasoning. They have proposed a method to discover inconsistencies in requirements automatically. This method used theorem proving and model checking techniques. These techniques were implemented in a prototype tool (CARL) with example. Alessandra et. al. [32] have presented an approach to identify and analyze inconsistencies and manage change. The approach is used to restructure requirements specification into parts through decomposition, and representing these parts into viewpoints. They have defined in viewpoints, and inter viewpoint rules for expressing specific relationship. The approach was evaluated through a case study.

3.3 Requirements Specification in Natural Language and Incompleteness

K. Wiegers said in his book [39] "Requirements are never finished or complete. There is no way to know for certain that you haven't overlooked some requirement, and there will always be some requirements that the analyst won't feel it

is necessary to record". Viktoria [17] have focus on to achieve requirement completeness. He has presented the importance of requirement incompleteness and its critical influence. In order to minimize the problem some requirements incompleteness symptom has been obtained to improve the project documentation and the quality of requirements. Sven et. al. [26] have presented an automated approach to improve the textual requirement specification using ontology to provide "common sense" for machine. The tool (RESI) marks certain word or sentence that is to be discussed with customer as many decisions cannot be made directly. This is a new approach and still negotiation is required. K. Wiegers notes [27] "Many software problems arise from shortcomings in the ways that people gather, document, agree on, and modify the product's requirements. The problem areas might include informal information gathering implied functionality, erroneous or uncommunicated assumptions, inadequately defined requirements and a casual change process". Requirements incompleteness has been address by other research. R. S. Carson [5] had suggested, describing system behavior in all possible condition rather than a particular situation. To justify any requirements and describes alternative action flows bring requirement consideration in respect to actor (i.e what actor is responsible for what functions). Another method recommends checking if requirements are for all system elements [33]. From the point of view of incompleteness source obtain method such approaches are quite one-sided. Method suggested in this paper includes above-mentioned approaches but considers them as special cases of incompleteness that are not enough. Obtaining incompleteness sources allows to research the problem more incompleteness and thus to assess requirements completeness more correctly.

3.4 Comparison of Alloy Formal Specification Language with other Formal Specification Languages

Although the most recommended method to address the inherited problem of natural language is formal specification, yet the most acceptable language used for writing requirements in industries is natural language despite of their inherited

problems. In this section we have done comparative study of alloy formal specification language with other formal specification language. Alloy is a declarative specification language for expressing complex structural constraints and behavior in a software system. Alloy provides a simple structural modeling tool based on first-order logic [24]. Alloy provides automatic analysis of requirement specification, alloy analyzer provide a model that satisfies logical formula written in alloy. Alloy have the ability of incremental analysis, the programmer may explore design idea by starting from a tiny model which is then scale up with alloy they are able to analysis it at every step [18]. Comparatively alloy have a conceptual simplicity and minimalism (mean very little to learn) and have no special semantics. Alloy have high level notations, constraint can be build incrementally, relations flexible and powerful and animating requirements [19]. Other formal requirement specification languages are VDM, OCL and Z-notation [21]. Both Alloy and VDM support object orientation and concurrency but VDM do not provide fully automatic analysis. OCL is implementation oriented and based on first order logic but uses syntax similar to programming language. OCL allows mixing declarative and operational elements. Contrary to OCL alloy has more conventional syntax and simpler semantic and is fully declarative. Z-notations have distinct style and notations of schema calculus that gives it the ability to support many different idioms. One of the advantages Z-Notations have that it has a rich mathematical notations making it more expressive than alloy but it automated only up to a point. Comparatively alloy to Z-notation, alloy has a pure ASCII notations and require no special typesetting tools. Witting out schema can be tedious in comparison to typing a signature in alloy. Theorem provers are more limited as compare to alloy analyzer.

3.5 Summary

The chapter presenting the relevant literature describes the tool, techniques and approaches that had address the inherited problem of natural language. Drawbacks of the existing approaches has discusses. We have also discussed the most recommenced method that address the inherited problems of natural language and there comparatives study with alloy formal specification language.

Chapter 4
Research Method

Chapter 4

Research Method

This chapter describes research method that has used to address the problem. The most appropriate method to address the problem investigated in this thesis is controlled experiment. Section 4.1 describes the rationality of the controlled experiment; section 4.2 and 4.3 present title and authorshipe of the method. section 4.4 describe method structure abstract. section 4.5 present keyword and 4.6 describes introduction to method. section 4.7 present detail planing of the experiment and section 4.8 present detail execution. section 4.9 describe detail of analysis procedure that how to analyse data. section 4.10 present the discussion of the data after analyis. section 4.11 present detail of conclusion and future. section 4.12 and 4.13 present acknowledgements and references respectively.

4.1 Research Method

Most appropriate method for this research is controlled experiment. Our techniques should be implemented in a real world setting and the other empirical method such as survey, mathematical modeling and case study is not suitable in this context. To answer the given research question I have proposed a two step planning. In the first step I will collect, review literature and propose a rigid format. Requirements should be written according to the proposed format.

Guidelines and recommendations should be provided to suggest how to write the requirements. In the second step requirement written according to the established format should be transformed into alloy specification. Two forms will be produced for alloy specification a mathematical form based on standard alloy specification and a textual form that uses a reduced set of English words to describe the systems. We will conduct controlled experiment with Graduate student of software engineering department selected randomly to validate our techniques. Prior to the execution of experiment, training should be given to the participant regarding the novel approach. Requirement specification written in natural language document should be used as a treatment material. Data will be collected and statistical method will be applied for analysis. Effectiveness of the novel approach with respect to detect ambiguity, inconsistencies and incompleteness in the requirements specification should be observed.

4.2 Title

Controlled Experiment

4.3 Authorship and Contact

Kiramat Rahman , Email: Shanglapk@gmail.com

4.4 Structured Abstract

4.4.1 Background

Despite of inherited problems of natural language it is the most acceptable method in industries for writing requirements specifications. Formal specification is the most recommended approach to address these issues but stakeholders do not understand requirements written in formal specification.

4.4.2 Objectives

We examined our navel approach with regard to detect inherited problem of natural language and provide support to stakeholder and developer.

4.4.3 Methods

We will conduct a controlled experiment with 18 randomly assign undergraduate student. The data will be collected using a questionnaire and analyzing using descriptive statistics and ANOVA test for testing Null Hypothesis.

4.4.4 Results

We expect the following result of the proposed approach. Automatic support for the analysis of formal requirement specification. An integrated approach that Support both stakeholder and Developer Minimize the inherited Problems of Natural language. An approach that Produce Accessible and Precise specification.

4.4.5 Limitation

Generalization of results is limited due to the fact that undergraduate students participated in the study

4.4.6 Conclusion

On the bases of expected result we conclude that our approach will be an effective for writing requirement specification either formally or informally. They will produce cost effective and quality product by minimizing the inherited problem of natural language and get in the benefits of formal specification as well

4.5 Keywords

Requirement specification document,Experimental study.

4.6 Introduction

4.6.1 Problem Statement

Informal requirement specifications are ambiguous, inconsistent and are incomplete. The stakeholder doesn't know about these problems while he writes requirements in natural language. These issues have to be observed by the requirement expert and it is important to address these issues. In order to develop system, according to the expectation, needs and requirements of stakeholder.

4.6.2 Research Objectives

Our main objectives is that without excluding natural or formal language despite of their weakness but convert these weakness into strengthen. Furthermore that is weakness of natural language that is ambiguity, inconsistency and incompleteness is address by formal language and the weakness of formal language that is complexity is achieved by natural language. Using of informal and formal language for writing requirement specification we can achieve both preciseness and accessibility. The approach should be cost effective in term of effort and time and lead to produce a quality product according to need and expectation of stakeholder

4.6.3 Context

Graduate student, incremental processes, tool for automatic analysis of formal specification.

4.6.4 Related Work

Previous work on informal specification and formal specification focus on particular issues. Either they have to achieve the accessibility of the specification or preciseness of the specification. our approach with respect advocate to achieve these two objectives and follow both method of informal specification and formal specification.

4.7 Experiment Planning

4.7.1 Goals

Informal requirements specification will be used to achieve the objective we will involve 8th term student of software engineering department those are more knowledgeable with respect to requirement specification and are passing through this processes by doing their final thesis

4.7.2 Experimental Units

8th term Graduate student of software engineering. Total participant will be 18 students. We will divide it into three team selected randomly.

4.7.3 Experimental Material

Informal requirement specification

4.7.4 Tasks

The student will use the techniques to generate formal specification from informal specification. Observe the techniques how much is that useful with respect to ambiguity, inconsistency and incompleteness identification. A questionnaire will be given to them to record their observation.

4.7.5 Hypotheses, Parameters, and Variables

Hypothesis:
H1: The techniques are effective in finding ambiguity, inconsistency and incompleteness in requirement specification.
H0: The techniques are not effective in finding ambiguity, inconsistency and incompleteness in requirement specification
Independent variable: Novel Approach Dependent Variable: unambiguous specification, consistent specification, complete specification

4.7.6 Experiment Design

Blocking experimental design has been chosen for this experiment. Blocking is the arrangement of experiment units in groups (blocks) that are similar to one another. We arranging three groups of student and all are similar to one other as they are 8th term graduate student.

4.7.7 Procedure

Experiment will be performed in software engineering lab and data will be collected through questionnaire. Tool that is used for automatic analysis of formal specification will be installed in the lab. Requirements specification will be used as treatment material.

4.7.8 Analysis Procedure

To analyze the data descriptive statistics will be used and ANOVA test will be used, because we are comparing different group and determining their mean is equal or not.

4.8 Execution

4.8.1 Preparation

Total 18 students will be selected. These 18 students will be dividing into three groups; each group will be consisting of six students. Participant will be trained with novel approach.

4.8.2 Deviations

One day will be given to participant to perform the task. Any deviation if occurs in the collection processes of data will be resolved.

4.9 Analysis

4.9.1 Descriptive Statistics

We will use a descriptive statics including number of observations, measures for central tendency, and dispersion. Mean, median, and mode will be used for measures of central tendency. Standard deviation, variance, range, as well as interval of variation and frequency will use for measures of dispersion.

4.9.2 Data Set Preparation

The preparation of the data set as a consequence of the descriptive statistics should be discussed. This includes data transformation, outlier identification and removal, and handling of Missing values as well as the discussion of drop outs.

4.9.3 Hypothesis Testing

For each hypothesis, quantitative results should be presented. If a null hypothesis is rejected, it has to be described on which significance level. Furthermore for individual performance, group performance investigation, a separate subsection for each analysis shall be used.

4.10 Discussion

4.10.1 Evaluation of Results and Implications

The results will be interpreted and detail discussion will be performed. Finding and observation will be discussed in detailn.

4.10.2 Threats to Validity

We will ensure internal and external validity to our experimental results. How is validity of the experimental results assured? How was the data actually validated? Threats that might have an impact on the validity of the results as such (threats

to internal validity, e.g., confounding variables, bias), and, furthermore, on the extent to which the hypothesis captures the objectives and the generalizability of the findings (threats to external validity, e.g., participants, materials) have to be discussed.

4.10.3 Inferences

Inferences statistics will be used to drawn inferences from the data to more general conditions.

4.10.4 Lessons Learned

What has been learned during the conduction of experiment has been discusse.

4.11 Conclusions and Future Work

4.11.1 Summary

In this section we will provide a concise Summary of the research and its results as presented in the Former sections.

4.11.2 Impact

We will provide the Description of impacts with regard to cost, schedule, and quality, circumstances under which the approach presumably will not yield the expected benefit

4.11.3 Future Work

We will recommend what other experiments could be run to further investigate the results yielded or evolve the Body of Knowledge.In this work our foucus is Alloy a spcification language we can done this experiment with other formal specification language to check the effectiveness of the idea.

4.11.4 Appendices

I will provide material, raw data, and detailed analyses, which might be helpful for others to build upon the reported work should be provided.

4.12 Acknowledgements

In this section ws will menstion participants, and (research) contributors who do not fulfill the requirements for authorship should be mentioned.

4.13 References

In this section, all cited literature has to be presented in the standard IEEE format.

Chapter 5
Proposed Solution

Chapter 5

Proposed Solution

In this chapter we have described proposed solution in detail. The chapter is organized as in section 5.1 problem statement are describes, section 5.2 have discussed the importance of the problem address, section 5.3 describe the overview of the proposed system. In Section 5.4 and subsequent subsection, Techniques has been presented.

5.1 Problem Statement

After the literature survey, we conclude that natural language is the common way to write requirement specification despite of their inherited problems that is ambiguity, inconsistencies and incompleteness. These issues cause unconscious misinterpretation of the requirements, which leads to develop a product that is not according to the need and expectations of stakeholder. Furthermore the most recommended method to address these issues is formal specification but the stakeholders do not understand the meaning and intention of formal specification. Additionally technical people are unable to negotiate requirements with stakeholders written in formal specification.

5.2 Problem Significance

Most recommended method for writing requirement is natural language despite of their associated inherited problems. It is important to address these issues at initial stage latter on you will pay huge amount to solve it. Worthwhile importances of the problem address in this dissertation are:

- Save time and money, comparatively it will take more time and spending more money to address it at latter stage.
- Producing a high quality product, improving requirement completeness, detecting ambiguity and inconsistency will lead to a product, which is to be of high quality.
- Decreasing communication gap between stakeholders and developer by presenting an approach that minimize this gap.

5.3 System Overview

In this task we have planned to propose techniques that addressing inherited problems of natural language through alloy specification. Alloy specifications are produced from textual user requirements written in English. This transformation is proposed to occur in two steps. In the first step, textual requirements written in any form are rearranged according to proposed format. Guidelines and recommendations have provided to suggest how to write the requirements. In the second step transformation rules are defined to transformed user requirements into alloy specification. We have provided two modes of description for Alloy specifications:

1. A mathematical form based on the standard Alloy specification language
2. A textual form that uses a reduced set of English words to describe the systems.

These two forms, that are equivalent, should permit to easily link the Alloy specifications to the textual requirements. We can either write the Alloy specifications or check if the corresponding sentences are part of the requirements or

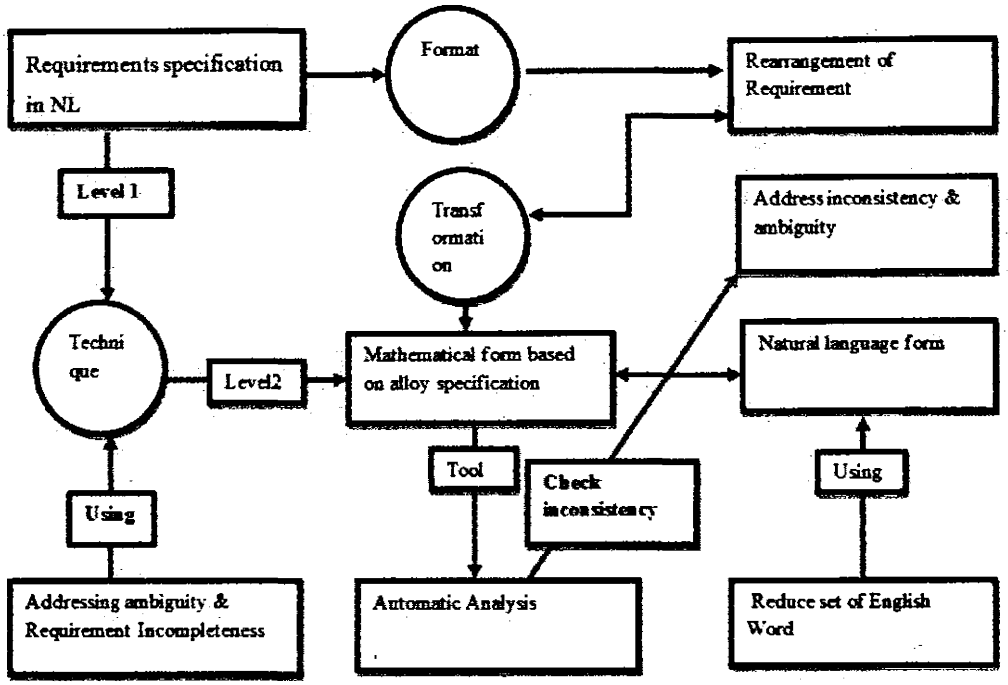


Figure 5.1: Overview of the Proposed Approach

alternatively write the sentences and obtain the Alloy specifications. Overview of the system has been presented in figure 5.1

5.4 Proposed Approach

We propose a two step approach for addressing the problem. The first step explained in Section 5.5 requires to re-arrange natural language requirements and guidelines hav been provided how to write requirements. The second step explained in Section 5.6 proposes techniques to generate alloy specification. Two forms have been provided for this specification. The two forms generated are semantically similar one is mathematical based on alloy specification language and the other is textual form using reduced set of English word to describe the system. These two forms are traceable to each other, we can write alloy specification and produce natural language statement of requirements or vice versa. A technique has proposed to address inconsistency in requirements. To address incompleteness

and ambiguity we have proposed techniques that address these issues at two levels that is informal and formal level. At informal level we have defined a dictionary of weak phrases that helps us to address issues and provide us a quantitative measure of the requirement to the extent that is ambiguous and incomplete. At level 2 Ambiguity will be detected through the automatic analysis of alloy specification using alloy analyzer. Alloy specification will provide a single interpretation of requirements. Traceability of alloy specification with textural form and textual form to alloy specification will help us to improve completeness.

5.5 Requirements Rearrangement Format

Natural language requirement should be re arranged according to this format. We have presented a template which provides a base for formal specification. Alloy is a model base approach that represents system requirements as system state model. We have modularized system requirements into state, relation and operation. The aim of modularization provides a single structure to represent different structure and mean of representing system requirements. State is a collection of named components and by extension we can add new component, Declaration and definition of invariant and derived component respectively. System is describes as a system state model. These requirements are extracted from Requirements written in natural language and will be present in the format shown in Figure 5.2

1. To restructure requirements written in natural language and represent system function define data invariant, state and operation.
2. Rearrange requirement according to the proposed format.
3. These steps form the basis for the transformation of natural language requirement into alloy specification.
4. the extracted requirements should be arrange according to the following format.

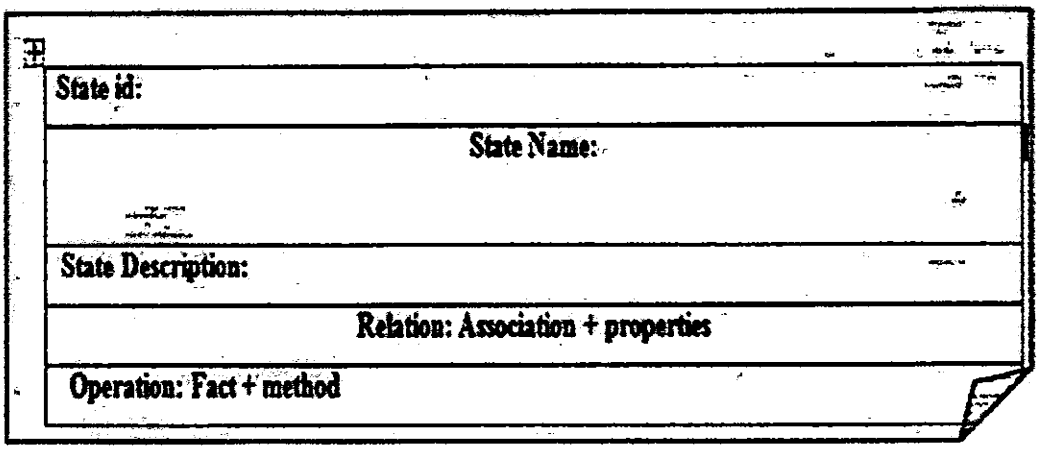


Figure 5.2: Requirements Rearrangement Format

5.5.1 Guide lines for writing Requirements

- Apply the above step to extract the information from the requirement written in Natural language.
- Rearrange the requirements according to the proposed format.
- State id: it is a unique id given to the state. Through these ids you can trace the requirements.
- State Name: it is meaning full name given to state. State is a collection of named components and by extension we can add new component. Declaration and definition of invariant and derived component respectively. System is describes as a system state model.
- State Description: this is brief description of the state, should be written.
- Relation: Describe the properties of the state and the association of it.
- Operation: Describes the methods that describe the system state and system transition and also the condition that is hold to be true.

5.6 Generating Alloy Specification

We have defined techniques that will generate alloy specification; techniques will provide two similar mood of alloy specification. Mathematical form based on standard alloy specification language which is to be analyzed through a tool (alloy analyzer) for the detection of ambiguity in the requirements. The other forms is base on reduce set of English words. These two forms are semantically equivalent and easily link alloy specification to the textural requirements and vice versa. This will help in improving completeness of requirement and detect inconsistencies in requirements. General Transformation Rule has been defined that will transform requirements into alloy specification.

- Formalize requirements state, invariant and operation according to the format presented in section 5.5
- Define Module that will encapsulate the alloy specification.
- Define Signature for state of system and declare respective relation and multiplicity in the body of signature.
- Specify Association and attribute as a relation in signature
- Define Predicate to represent operation that has effect on state along with pre and post condition.
- Instantiate the model and generate respective instances with given scope.
- Develop the specification incrementally by checking assertion through alloy analyzer and refine the specification.
- Produce two forms for alloy specification.
- Mathematical form (based on alloy)
- Natural language form (used consistent Requirements language)
- These two forms are traceable to each other.

Module	Module is used to structure and encapsulate alloy specification. it consist of the key word module and name of the module				
	Sig name {Decl}	Alloy specification start with defining signature (is set of atoms) used throughout the specification			
		Fact name{}	That is a predicate /constraint that hold in all instances of alloy specification		
			Pred name{}	Consist of one or more constraints and can be used to represent operation.	
An expression that return a results.				Fun Name{}	
Predicate that visualize the alloy specification				Pred Show()	
Command to ask Alloy to find instances that satisfy a predicate.					Run show()

Figure 5.3: alloy specification structure

5.6.1 Mathematical Forms

A mathematical form of requirement specification is base on formal specification language alloy. This is a form that represents system specification as a system state model. This model is constructed using alloy which is based on mathematical concept such as sets and function. It is executable and amenable to automatic analysis. Alloy analyzer is a standard tool used to find a counter example of the model produce through these techniques. The tool support alloy specification to find ambiguity in requirements. Alloy specification general structure has been shown in figure 5.3

5.6.2 Textual Form

This is another form of alloy specification used reduce set of English words to describes system. These two forms, that are equivalent, should permit to easily link the Alloy specifications to the textual requirements. We can either write

the Alloy specifications or check if the corresponding sentences are part of the requirements or alternatively write the sentences and obtain the Alloy specifications. To Write requirement in natural language form we are using a combine approach that is using IEEE-830 Standard "Shall Statement" and using reduce set of English word define in the following dictionary. This dictionary provides a consistent language for writing requirement in natural language and reduces the inherited problem of natural language. Table 5.1 shows the word/phrases that we will use in our approach to represent requirement specification. These word/phrases command that something must be provided.

Table 5.1: Dictionary of Reduce set of English Words

Word	**Situation used**
Shall	"Shall" is used to write functional capability.
Must or must not	To write requirements performance or constraint used these words.
Is required to	is often used as an imperative in specifications statements written in the passive voice
Are applicable	When you are including other documentation, standard or by reference as an addition to the requirements already specified.
Responsible for	It is used for requirements that are written for systems whose architectures are already defined.
Will	Is generally used to cite things that the operational or development environments are to provide to the capability being specified. For example,"The building's electrical system will power the XYZ system."
Should	Is not frequently used as an imperative in requirement specification statements. However, when it is used, the specifications statement is always found to be very weak. For example, "Within reason, data files should have the same time span to facilitate ease of use and data comparison."

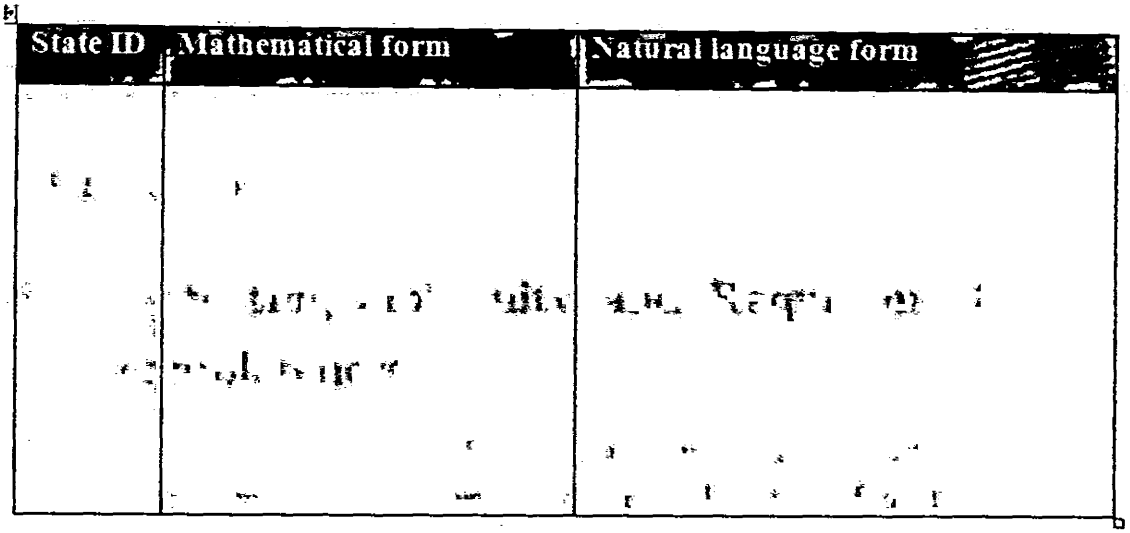


Figure 5.4: Requirements Traceability link

5.6.3 Traceability link among Requirement

The two form of alloy specification should be represent parallel to link with each other the following table and the unique id will help us to link the two form of requirements. The formate is given in Figure 5.4

5.7 Detecting and Removing Inconsistencies

Alloy specification allows us to check the inconsistencies of system model and additionally providing the scenario showing the reason of inconsistencies. We check the inconsistency through the appropriate using of two alloy statement fact and assert.

1. Execute alloy specification through tool alloy analyzer.
2. The tool will generate instances that represent the possible inconsistency in alloy specification.
3. If the model is inconsistent check your specifications.
4. Apply alloy statement fact to remove the inconsistency

State id: Queue_01
State Name: Queue
State Description: queue represents a set of queue. Queues have a single association of type node.
Property and multiplicity • Multiplicity is one or zero • Every node must belong to some node • Queue must belong
Operation: size( ), pop_front( ), pop_back()

Figure 6.1: State Format 1

State id: Node_02
State Name: node
State Description: every link list has a node that consists of data and next field with single multiplicity.
Property and multiplicity: • Multiplicity is One or zero • Node n must not call himself • Cycle must not exist between node
Operation: Insert node (), Delete node()

Figure 6.2: State Format 2

Alloy Specification	Tool Results	Finding

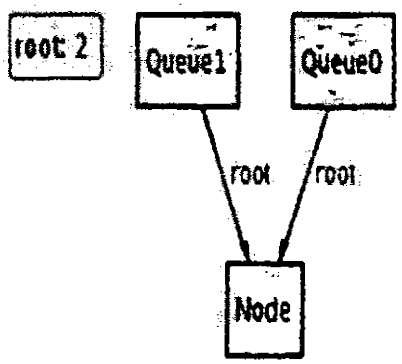
module examples/tutorial/Queue
sig Queue { root: Node }
sig Node { next: lone Node }
fact nextNotReflexive { no n:Node | n = n.next }
fact allNodesBelongToSomeQueue {
all n:Node | some q:Queue | n in q.root.next
}
Fact nextNotCyclic {no n:Node | n in n.^next }
pred show() {}
run show for 2
		We observe that the node belong to two different queue. So every node will belong to exactly one queue.

Figure 6.4: Incrment 2

Alloy Specification	Tool Result	Finding

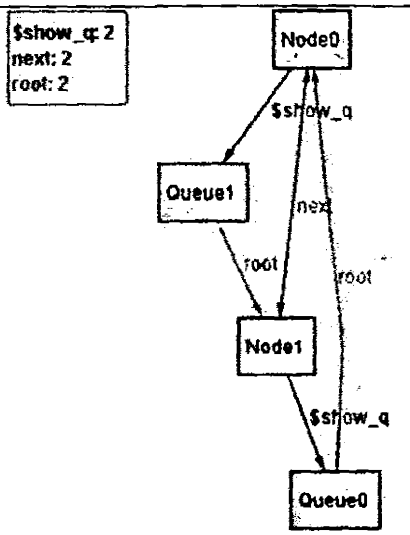
module examples/tutorial/Queue
sig Queue { root: Node }
sig Node { next: lone Node }
fact nextNotReflexive { no n:Node | n = n.next }
fact allNodesBelongToSomeQueue {
all n:Node | some q:Queue | n in q.root.next
}
pred show() {}
run show for 2
		During queue implementation no cycle exist between nodes. But the tool shows that cycle exist between nodes.

Figure 6.5: Incrment 3

Alloy Specification	Tool Results	Finding

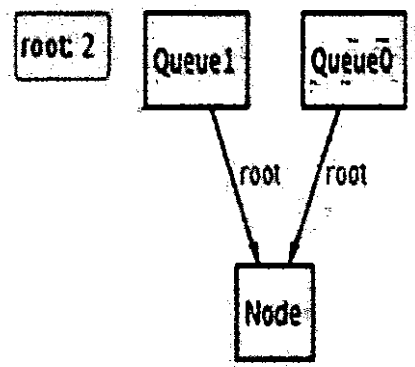
module examples/tutorial/Queue
sig Queue { root: Node }
sig Node { next: lone Node }
fact nextNotReflexive { no n:Node | n = n.next }
fact allNodesBelongToSomeQueue {
all n:Node | some q:Queue | n in q.root.next
}
Fact nextNotCyclic {no n:Node | n in n.^next }
pred show() {}
run show for 2
		We observe that the node belong to two different queue. So every node will belong to exactly one queue.

Figure 6.6: Incrment 4

Alloy specification	Tool Result	Finding

Moduleexamples/tutorial/Queue
sig Queue { root: Node }
sig Node { next: lone Node }
fact nextNotReflexive { no n:Node | n = n.next }
fact allNodesBelongToSomeQueue {
all n:Node | some q:Queue | n in q.root.next}
fact nextNotCyclic { no n:Node | n in n.^next }
fact allNodesBelongToOneQueue {
all n:Node | one q:Queue | n in q.root.*next
}
pred show(){}
run show for 2
	No more instances are found	We observe that we have not found no more specification is unambiguous, consistent and complete

Figure 6.7: Incrment 5

will gain confidence to discuss requirement with the owner of requirements for clarification and resolving any issue of incompletes, ambiguity and inconsistency.

- Queue shall be implemented as links list
- Link list shall require to have a queue
- Link list should have a node
- Node must not call himself
- Every link node must belong to some Queue.
- Cycle must not exist between nodes.
- Queue must exactly connected to one node

6.1.6 Traceability between two forms

Finding and observation with example1 in the previous example we have implemented our proposed approach; our approach is effective in term of finding ambiguity, inconsistencies and incompleteness in requirement written in natural language. Our approach have provide a confidence in using natural language and formal language at same time and increase understandability between stakeholder and technical people. traceability between two form are shwon in figure 6.8

6.2 Evaluation of Proposed Approach (Experiment)

We have evaluated our approach with 18 student of software engineering. The subjects were dividing into three groups selected randomly. Each team was given an informal requirement specification and asked to find out ambiguity, inconsistency and incompleteness using the new proposed techniques. The student had no prior knowledge of the number or nature of issues existing in the specification but they were simple told that "suspicion that the program is not perfect". Material and training were given to participant in advance. The purpose of this experiment is to evaluate the effectiveness of the proposed techniques.

State id	Mathematical form	Textual form
Queue_01, Node_02	module example/thesis/queue/liklist sig Queue {root: lone Node} sig Node {next: lone Node} fact nextnoreflexive { no n:nod|n=n.next} fact belongtosomequeue{ all n:node|some q.queue|n In q.root.*next} Fact nextnotcycle {No n:node|n in n.^next} Fact allnodebelongtoonequeue { all n:node|one q:queue|n in q.root.*next} pred show () {} run show for 2	• Queue shall be implemented as links list • Link list shall require to have a queue • Link list should have a node • Node must not call himself • Every link node must belong to some queue. • Cycle must not exist between nodes. • Queue must exactly connected to one node

Figure 6.8: Traceability between two forms

6.2.1 The Subjects

The 18 subjects were students in a course on requirement engineering and formal specification at international Islamic university. The student have including MS (Software Engineering) research Student and under graduate 8th term student who are working on his final degree thesis. Students have an average experience in writing requirement for their final thesis and have done so many assignments given to them in the last four years or some student who are working in software houses. Students were motivated through training given to them.

6.2.2 The Experimental Design

In design of experiment without biasness is a crucial part, we had performed some pretest on subject to avoid biasness. Pretest was performed through a questionnaire. We have asked from all participants though following questionnaire. Writing requirements specification experience was plot on scale of 1-5.(1= no experience, 2= writing specification infrequently, 3= writing specification frequently 4= primary job is to write requirement). The subject were also tested by giving them a simple informal specification and asked to find out ambiguity, inconsistency and incompleteness in requirements using the proposed approach.

6.2.3 Informal Requirements Specification

Informal requirement specification had written in natural language (English). Several inherited issues of natural language were seeded into informal specification. Each team was given this specification. The subjects were not told how many inherited issues exist in specification or the specification had problems. They were asked to find out inherited problem of natural language in requirement specification and keep track of them how many problem you have found. The informal specification that was used is as follow. "This is the minimum requirements for queue implementation. Link list is an effective implementation of abstract data structure. Queue is one of abstract data structure which is easy to implement through a link list. Link list have a root node and successor node. The queue implemented linked list with nodes and with a pointer to both the

?rst and last nodes of the list. Queue be able to insert node at the tail and head. In link list each node will be link with next node as appropriate. No link will be able to call himself, In normal scenario every node will belong to their parent queue. The total seeded issues in the treatment specification are nineteen which has been shown in the Tables 6.2 , 6.3 and 6.4 respectively. Requirments

Table 6.2: Known Ambiguity

Queue is one of abstract data structure which is easy to implement through a link list
In link list each node will be link with next node as appropriate
No link will be able to call himself
In normal scenario every node will belong to their parent queue
Minimum requirements for queue implementation
linked list have a nodes and a pointer to both the first and last nodes.
The queue implemented linked list with nodes
Queue be able to insert node at the tail and head.

Table 6.3: Know Incomplete Requirements

Add operation not explicitly define in informal requirements
Delete operation not explicitly define in informal requirements
in case of empty queue there is no error handling is stated
minimum requirements for queue implementation
The specification has not stated some non-functional behavior.
No precise specification for object change from state to state.

presented in table 6.4 is contradact with other requirments and constraints.

Table 6.4: Known Inconsistent Requirements

Operation type or not defined
Changing requirements
Every link node must belong to some queue
Node must exactly connected to one queue
Node must not call himself

6.2.4 Data Collection

Questionnaire has been distributed at the end of experiment asking each group to describes the ambiguities, inconsistency and incompleteness they have found in informal requirement using the proposed approach and the time spend, ease of learning, and effectiveness of tool in finding inherited problem, understandability of the requirement, communication gap decreases or not

6.2.5 Data Analysis

The purpose of the Data analysis is to summarize the data collected and the treatment of the data. Data has been analyzed according to the design of experiment. In have structure the analysis of data in to following section.

6.2.5.1 Descriptive statistics and Data Set Preparation

The purpose of this subsection is to present collected data statically by applying descriptive statics. We have implemented the appropriate statistics including, measure for central tendency, dispersion and number of observation. The consequence of descriptive statistics has been discussed and presented graphically. Participants Response to the Rating Scale Question in this section we are presenting the response of the participaint after the execution of the controlled experiment. Data was collected through questionnnaire.Statistical method was used to find more meaning and take and informed decision.

1. How would you rate the proposed approach? What participant thought of about the reliability and validity of the techniques? Users were given a five point rating scale (from superb to good and fair) Here are the responses from 18 users:
   5, 5, 5, 5, 4, 5, 3, 4, 5, 5, 5, 5, 4, 5, 1, 2, 3, 4
   Because the question was just written for the survey, there's no historical or comparative data. To find more meaning in this jumble of numbers mean and standard deviation were calculated as there were 18 responses and the mean was a 4.167 and the standard deviation a 1.21. Percent response of the 18 participant has been show in the Figure 6.9.

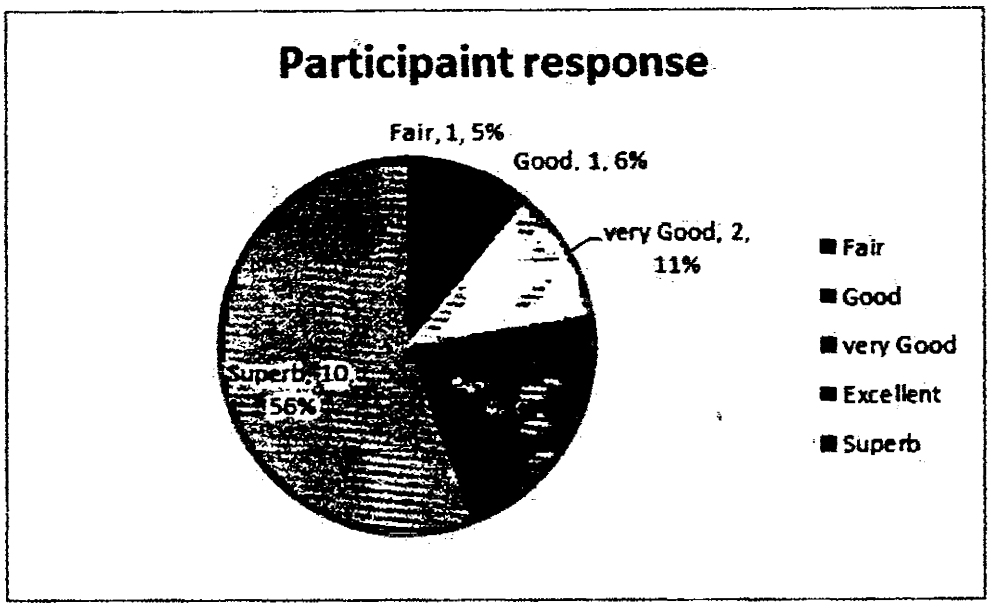


Figure 6.9: Response to Q 1

2. How would you rate the problem address are cost effective? Here are the responses from 18 users:
   5, 5, 5, 4, 5, 3, 4, 5, 5, 5, 5, 5, 5, 4, 3, 2, 2, 3
   Because the question was just written for the survey, there's no historical or comparative data. To find more meaning in this jumble of numbers mean and standard deviation were calculated as there were 18 responses and the mean was a 4.167 and the standard deviation a 1.20049 Percent response of the 18 participant has been show in the Figure 6.10.

3. Propose approach has improved Requirement completeness? Here are the responses from 18 users:
   5, 5, 5, 4, 5, 3, 4, 5, 5, 5, 5, 5, 5, 4, 3, 2, 2, 3
   Because the question was just written for the survey, there's no historical or comparative data. To find more meaning in this jumble of numbers mean and standard deviation were calculated as there were 18 responses and the mean was a 4.11 and the standard deviation a 1.182663 Percent response of the 18 participant has been show in the Figure 6.11

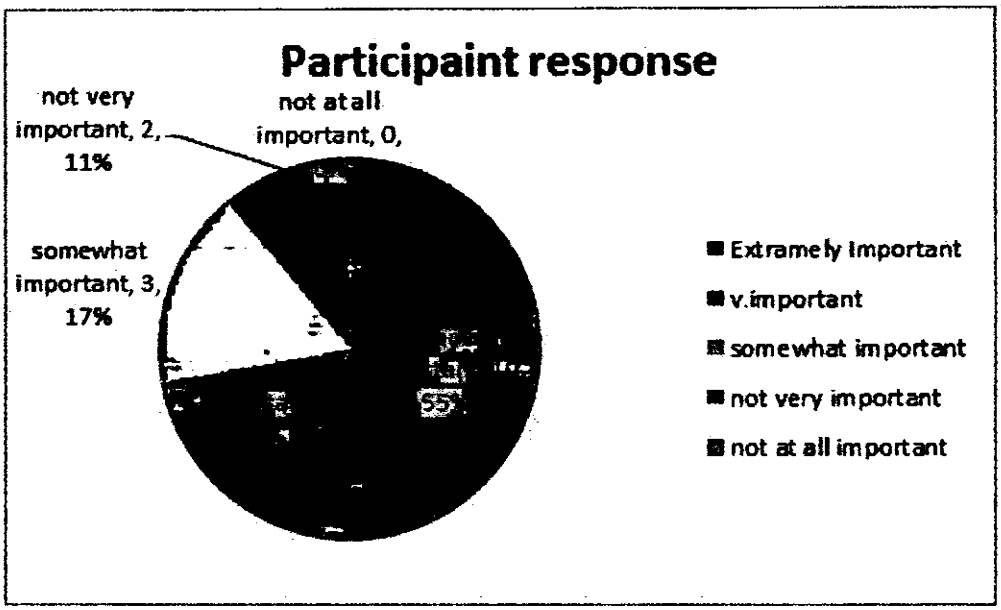


Figure 6.10: Response to Q 2

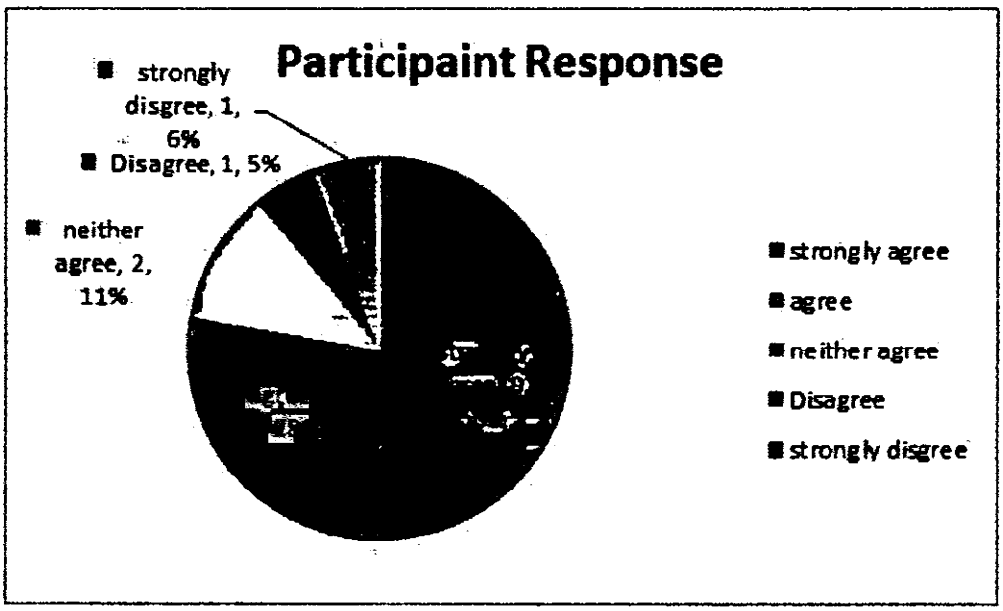


Figure 6.11: Response to Q 3

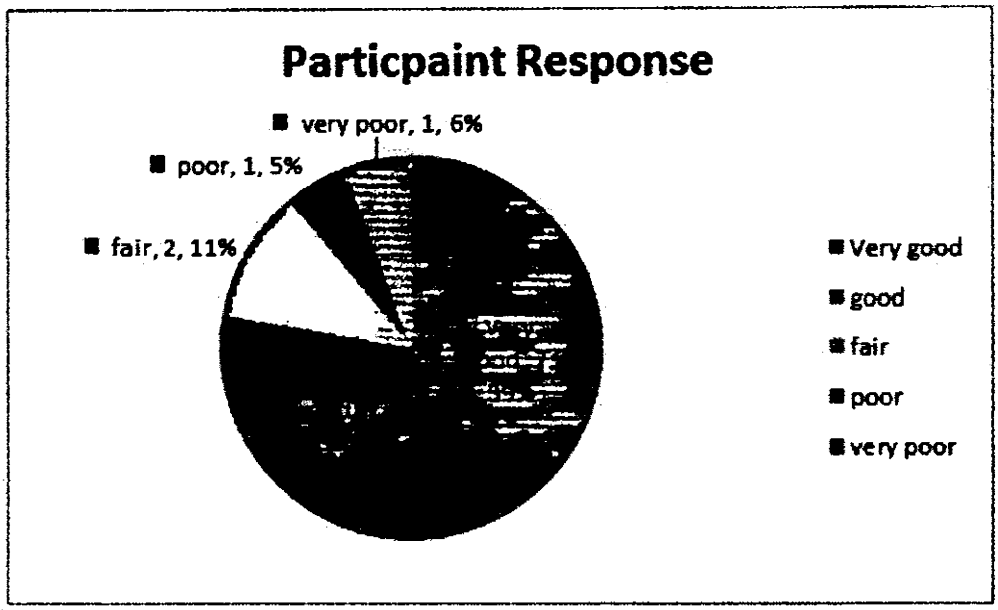


Figure 6.12: Response to Q 4

4. How would you rate the performance of the proposed approach?

Here are the responses from 18 users:
5, 4, 5, 5, 4, 4, 5, 4, 2, 5, 4, 1, 5, 4, 3, 5, 4, 3
Because the question was just written for the survey, there's no historical or comparative data. To find more meaning in this jumble of numbers mean and standard deviation were calculated as there were 18 responses and the mean was a 4.38 and the standard deviation a 0.697802 Percent response of the 18 participant has been show in the Figure 6.12
Data point extracted from the raw data of the controlled experiment that is the finding of ambiguity inconsistency and incompleteness by participants using the proposed approach in informal requirement specification. We have process the raw data and prepare the data point for testing our hypothesis. Table 6.5 shows data point. On the basis of these sample data point we have calculated individual mean, grand mean which is show in Table 6.6 On the basis of mean and grand mean we calculated the dispersion of data.
Sum of Square Total (SST) = SSB+ SSW

Table 6.5: Sample Point

Team A	Team B	Team C
5	5	3
6	3	2
7	4	1

Table 6.6: Central Tendency

	Team A	Team B	Team C
Individual Mean	6	4	2
Grand Mean	4		

1. Sum of Square between (SSB) groups Calculation: =

   $(6-4)^2+(6-4)^2+(6-4)^2+(4-4)^2+(4-4)^2+(4-4)^2+(2-4)^2+(2-4)^2+(2-4)^2$

   = 24

2. Sum of Square within (SSW) Group Calculation: =

   $(5-6)^2+(6-6)^2+(7-6)^2+(5-4)^2+(3-4)^2+(4-4)^2+(3-2)^2+(2-2)^2+(1-2)^2$

   = 6

3. Sum of Square Total Calculation: 24 + 6 = 30

4. SSB Degree of Freedom: n - 1 = 3 - 1 = 2

5. SSW Degree of Freedom: m(n - 1) = 3(3 -) = 6

6. SST Degree of Freedom: m.n - 1 = 9 - 1 = 8

6.2.5.2 Hypothesis testing

In this section we have evaluated the data and analysis of the proposed approach. Inferential statistics has been reported with value of the test, degree of certainty,

Table 6.7: ANOVA Table

Source of Variation	Sum of Squares	Degree of Freedom	Mean Squares	F-Statistics	F-table
Between Group	24	2	4	12	5.14
Within Group	6	6	3		

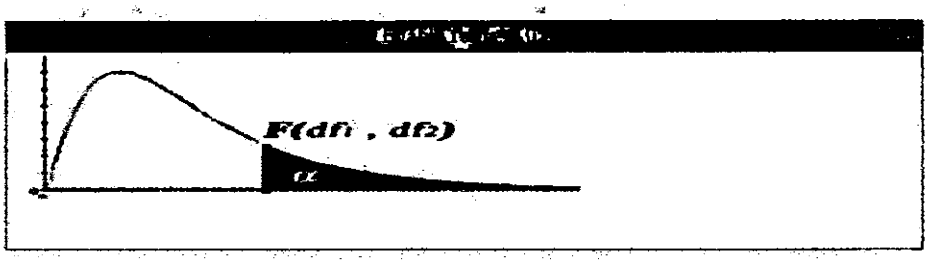


Figure 6.13: Right Skew Distribution

degree of freedom and F-statics values. The null hypothesis has rejected with a significance level. Further individual performance and tram performance has been discussed.

H0: The techniques are not effective in finding ambiguity, inconsistency and incompleteness in requirement
specification

H1: The techniques are effective in finding ambiguity, inconsistency and incompleteness in requirement specification.

Let assume that null hypothesis is true it means that $\mu 1 = \mu 2 = \mu 3$

If one of them is different than alternative hypothesis is true. The rejection of Null hypothesis does not mean that the acceptance of alternative hypothesis so we have to determine the degree of certainty require for ANOVA test. In current scenario we require 95% certainty so we have to calculate alpha = 1 - 95% = 0.05. In Table 6.7 and Figure 6.13 detail of ANOVA test are given.

F-statistics value is greater than F-table Value so we reject the Null Hypothesis accept the alternative hypothesis which is that the proposed technique is

effective in finding ambiguity, inconsistency, incompleteness in informal requirement specification.

6.2.6 Discussion

One of the basic results of the experiment was that the proposed approach is highly effective in finding ambiguity, inconsistency and incompleteness in requirements. The total nineteen seeded inherited problem in informal specification has presented to use the techniques to find out these problem. The experimental result shows remarkable results that average twelve inherited problem has detected by the three teams. Performance of individual and team has measured using descriptive statistics. Statistically with 95% significance level and alpha value 0.05. We have rejected the null hypothesis and accept the alternative hypothesis. Data collected from the participant shows that the presented approach is efficient and effective in finding inherited problems of natural language. The issue address is cost effective because the efforts spend in detecting the inherited problems using the techniques are low. The approach has a negative impact that is still needed to learn formal specification language so the training is required while using this approach. The approach is novel so it will take time to get mature. The possible criticism of the experiment is the small size of informal specification used in the experiment, yet we believe that it should be applicable for large informal specification

6.3 Summary

This is the most important chapter of the thesis, presenting detail implementation of the proposed approach with example. Controlled experiment has been executed in the lab environment and data collected from the participant through questionnaire. The raw data has been processes and descriptive statistic has been implemented and have find out mean and dispersion of data. ANOVA test were applied to test the hypothesis with a 95% significance level, degree of freedom and alpha of 0.05, and finally the result has been discusses and interpreted in discussion section.

Chapter 7
Conclusion & Future work

Chapter 7

Conclusions and Future Work

In this chapter we briefly review the thesis, concluding our finding and observation. Section 7.1 describes conclusion and section 7.2 present future works.

7.1 Conclusions

In software world requirements is the key of success. Software develops for the people by the professionals. In software industries the most common and acceptable language of writing requirements specification is natural language. The reason is that the stakeholders understand the language which they speak and write requirements in that. Requirements written in natural language have inherited problem of ambiguity, incompleteness and inconsistency. Furthermore the most recommended method to address these issues is formal method or writing formal specification. Formal method is based on mathematical concept and logic which is not understandable for non technical people.

Literature was investigated to address these issues but no integrated approach was found to address these issues. In this dissertation, we have proposed an integrated approach that addresses these issues. This approach is base on the idea to write requirements specification in way to achieve both accessibility and preciseness and also address the inherited problem of natural language. To achieve

this we have used alloy specification language and reduce set of English word and has proposed techniques to address the issues of requirements written in alloy.

A two step approach has been offered; in first step a format has been proposed. Informal requirements specification should be rearrange according to this format. Guide lines are provided how to write requirements. This is an intermediate step for transformation of informal requirement into formal requirement. One of our finding is that proposed format will produced a modular requirements specification which makes change easy in requirements and its propagation and impact on other requirements.

A transformation rule has been defined to generate alloy specification. Two form of alloy specification has been produced; One is mathematical form which is base on alloy specification language and the other is textual form using a reduce set of English word. These two forms are equivalent semantically and are traceable to each other. We have proposed a traceability link to make traceable these two forms.

Technique has been proposed to address inconsistency in the requirement using alloy specification. We are using two statements of alloy that are fact and assertion to detect inconsistency in requirements.

The techniques proposed for detecting ambiguity and incompleteness address it at two levels, Informal level and formal level. Additionally, the reason of addressing ambiguity and requirements incompleteness, at informal level is, if the requirement is ambiguous and incomplete from stakeholder then it is difficult to address it through formal specification. We have observed that the proposed technique is cost effective in finding these two issues.

The Proposed approach has been explained with example. We implement it through real case scenario. Controlled experiment was conducted to evaluate the propose approach. Informal requirement specification with known seeded ambiguity, incompleteness and inconsistency were used as a treatment material. The participants were trained and Pre-test was performed to reduce biasness and insured internal validity, after the execution of experiment data was collected through data collection questionnaire. We have processes raw data and come to the conclusion that the proposed technique is effective in finding inherited problems of natural language. Null hypothesis were rejected with 95% significance

level and it has been proved statistically.

The problem address in this thesis has been resolved through our proposed approach and proved it experimentally. Our observation and finding of the proposed approach is that this approach support both stakeholder and technical people and help in understanding the requirements unambiguously. The approach support to improve requirements completeness and resolve inconsistency found in requirements.

The approach has high impact on producing a quality specification which leads to the development of software according to need and expectation of stakeholder. Comprehensive, complete and inconsistent requirements have an impact on quality of the ultimate product. The cost of addressing these issues at early stage is low as compared to latter stage and the proposed approach is effective in finding these issues at early stages. The time and effort require to use the technique is low. We can acchieve grate success with less effort and time by using this technique.

7.2 Limitation & Future Work

There is always a scope for improving. Future work is to solve the limitation of the existing approaches.

- **Automatic Transformation**

  We have defined transformation rule which transform textual requirement into formal alloy specification. This work is done manually it is possible to automate it. This will decrease effort and save time.

- **Implementation with other formal language**

  A single language is not fit for all types of specification so the techniques proposed here may implement with other formal specification language.

- **Implementation in Industrial case study**

  Validation of the proposed approach may be validated though implementing it with an industrial case study.

- **Complete automation**

  The approach is semi automated it is possible to develop an integrated tool to fully automate the approach.

Appdx A

Pre Test Form writing requirements specification experience help us by completing this short questionnaire. Indicate your level of experience in writing requirements specification. Please answer each question honestly and to the best of your ability. We will use your feedback to determine how we can rate our experiment and it has an impact on our research. Please Plot your experience on scale of 1-5.

1. no experience
2. writing specification infrequently
3. writing specification frequently
4. primary job is to write requirement
5. expert of writing requirement specification

Appdx B

Data Collection Questionnaire Please help us by completing this short questionnaire. Please answer each question honestly and to the best of your ability. We will use your feedback to determine how we can rate our proposed approach.

- How would you rate the proposed approach?

  1. Fair
  2. good
  3. Excellent
  4. Superb

  Note: Rating on the basis of reliability and validity of the techniques

- How would you rate the problem address are cost effective

  1. Extremely
  2. important
  3. Very Important
  4. Somewhat Important
  5. not very important Not at all important

  Note: cost effective in term of effort and time

- Propose approach has improved Requirement completeness?

  1. Strongly Agree
  2. Agree

3. Neither Agree
4. Disagree
5. Strongly Disagree

- How would you rate the performance of the proposed approach?

1. Very Good
2. Good
3. Fair
4. Poor
5. Very poor

Note: based on the achievements of target by the techniques These question to be filled by Teams

- What are the numbers of inconsistent requirements you have detected?
- What are the numbers of ambiguity you have detected in requirements?
- What are number of incomplete requirements you have detected?

Bibliography

[1] Ieee recommended practice for software requirements specifications. *IEEE Std 830-1998*, page i, 1998. Cited on page:1.

[2] I. Alexander and L. Beus-Dukic. *Discovering Requirements: How to Specify Products and Services.* John Wiley & Sons, 2009. Cited on page:13.

[3] V. Ambriola and V. Gervasi. Processing natural language requirements. In *Proceedings of the 12th international conference on Automated software engineering (formerly: KBSE)*, ASE '97, pages 36–, Washington, DC, USA, 1997. IEEE Computer Society. Cited on page:14.

[4] Robert Balzer. Tolerating inconsistency. In *Proceedings of the 13th international conference on Software engineering*, ICSE '91, pages 158–165, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press. Cited on page:16.

[5] Ronald S Carson. Requirements completeness: A deterministic approach. In *Proceedings of the INCOSE*, pages 739–746., 1998. Cited on page:17.

[6] F. Chantree, A. Kilgarriff, A. de Roeck, and A. Willis. Disambiguating coordinations using word distribution information. In *In Proceedings of Recent Advances in Natural Language Processing (RANLP)*, pages 144–151, Borovets, Bulgaria, 2005. Cited on page:13.

[7] Francis Chantree, Bashar Nuseibeh, Anne de Roeck, and Alistair Willis. Identifying nocuous ambiguities in natural language requirements. In *Proceedings of the 14th IEEE International Requirements*

Engineering Conference, RE '06, pages 56–65, Washington, DC, USA, 2006. IEEE Computer Society. Cited on page:14.

[8] KOEN CLAESSEN. New techniques that improve mace-style finite model finding. In *Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications*, 2003. Cited on page:6.

[9] ALAN M. DAVIS. *Software requirements: objects, functions, and states.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. Cited on page:1.

[10] A.P.G. EBERLEIN. *Requirements Acquisition and Specification for Telecommunication Services.* University of Wales Swansea, 1997. Cited on page:9.

[11] F. FABBRINI, M. FUSANI, V. GERVASI, S. GNESI, AND S. RUGGIERI. On llnguistic quality of natural language requirements, 1998. Cited on page:8.

[12] ALESSANDRO FANTECHI, STEFANIA GNESI, G. LAMI, AND A. MACCARI. Application of linguistic techniques for use case analysis. In *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, RE '02, pages 157–164, Washington, DC, USA, 2002. IEEE Computer Society. Cited on page:14.

[13] DOV M. GABBAY AND ANTHONY HUNTER. Making inconsistency respectable: a logical framework for inconsistency in reasoning. In *Proceedings of the International Workshop on Fundamentals of Artificial Intelligence Research*, FAIR '91, pages 19–32, London, UK, UK, 1991. Springer-Verlag. Cited on page:16.

[14] DONALD C. GAUSE AND GERALD M WEINBERG. *Exploring Requirements: Quality Before Design.* Dorset House Publishing Company, Incorporated, 1989. Cited on page:2.

[15] VINCENZO GERVASI AND DIDAR ZOWGHI. Reasoning about inconsistencies in natural language requirements. *ACM Trans. Softw. Eng. Methodol.*, 14[3]:277–330, July 2005. Cited on page:16.

[16] C. GHEZZI AND B. NUSEIBEH. Guest editorial: introduction to the special section. *Software Engineering, IEEE Transactions on*, **25**[6]:782 –783, nov/dec 1999. Cited on page:16.

[17] VIKTORIA GINGINA. On requirements completeness analysis method. In *Proceedings of the 4th Spring/Summer Young Researchers Colloquium on Software Engineering SYRCoSE*, pages 29–36., 2010. Cited on page:17.

[18] DANIEL JACKSON. A comparison of object modelling notations: Alloy, uml and z. Technical report, 1999. Cited on page:18.

[19] DANIEL JACKSON. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, **11**[2]:256–290, April 2002. Cited on page:18.

[20] DANIEL JACKSON. *Software Abstractions: Logic, Language, and Analysis.* The MIT Press, 2006. Cited on page:5.

[21] JEREMY L. JACOB. Trace specifications in alloy. In *Proceedings of the Second international conference on Abstract State Machines, Alloy, B and Z*, ABZ'10, pages 105–117, Berlin, Heidelberg, 2010. Springer-Verlag. Cited on page:18.

[22] E. KAMSTIES AND B. PAECH. Taming ambiguity in natural language requirements. In *in Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications*, Paris, France, 2000. Cited on Pages:2 and 14.

[23] ERIK KAMSTIES, DANIEL M. BERRY, BARBARA PAECH, E. KAMSTIES, D. M. BERRY, AND B. PAECH. Detecting ambiguities in requirements documents using inspections. In *in Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01*, pages 68–80, 2001. Cited on Pages:2 and 13.

[24] SARFRAZ KHURSHID, MUHAMMAD ZUBAIR MALIK, AND ENGIN UZUNCAOVA. An automated approach for writing alloy specifications using instances. In *Proceedings of the Second International Symposium on Lever-*

aging Applications of Formal Methods, Verification and Validation, ISOLA '06, pages 449–457, Washington, DC, USA, 2006. IEEE Computer Society. Cited on page:18.

[25] NADZEYA KIYAVITSKAYA, NICOLA ZENI, LUISA MICH, AND DANIEL M. BERRY. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requirements Engineering*, **13**[3]:207–239, August 2008. Cited on page:14.

[26] SVEN J. KORNER AND TORBEN BRUMM. Resi - a natural language specification improver. *International Conference on Semantic Computing*, **0**:1–8, 2009. Cited on page:17.

[27] K.WIEGERS. *Software Requirements: Practical Techniques for Gathering and Managing Requirements throughout the Product Development Cycle.* Redmond, Wash: Microsoft Press,, 2nd edition edition, 2003. Cited on page:17.

[28] A. MAVIN, P. WILKINSON, A. HARWOOD, AND M. NOVAK. Easy approach to requirements syntax (ears). In *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, pages 317 –322, 31 2009-sept. 4 2009. Cited on page:14.

[29] R.S. PRESSMAN. *in Software Engineering A Practitioners Approach*, chapter analysis concepts and principles, pages 673–694. McGraw Hill 0-07-118458-9, 5ed edition, 2001. Cited on page:8.

[30] R.S. PRESSMAN. *in Software Engineering A Practitioners Approach*, chapter Formal Methods, pages 673–694. McGraw Hill 0-07-118458-9, 5ed edition, 2001. Cited on page:10.

[31] R.S. PRESSMAN. *Software Engineering A Practitioners Approach,*, chapter system engineering, pages 258–259. McGraw Hill, 0-07-118458-9, 5ed edition, 2001. Cited on page:9.

[32] ALESSANDRA RUSSO, BASHAR NUSEIBEH, AND JEFF KRAMER. Restructuring requirements specifications for inconsistency analysis: A case study.

In *Third Intl Conf. Requirements Engineering, IEEE CS Press, Los Alamitos, Calif*, pages 51–60. IEEE Computer Society Press, 1998. Cited on page:16.

[33] J. ROBERTSON S. ROBERTSON. *Mastering the Requirements Process.* London: Addison Wesley Professional., second edition edition, 2006. Cited on page:17.

[34] KOWALSKI ROBERT. SADRI, FARIBA. *An application of general purpose theorem-proving to database integrity.* London : University of London, Imperial College of Science and Technology, Dept. of Computing,, 1986. Cited on page:16.

[35] G. MICHAEL SCHNEIDER, JOHNNY MARTIN, AND W. T. TSAI. An experimental study of fault detection in user requirements documents. *ACM Trans. Softw. Eng. Methodol.*, 1[2]:188–204, April 1992. Cited on page:1.

[36] R. L. SMITH, G. S. AVRUNIN, AND L. CLARKE. From natural language requirements to rigorous property specifications. In *Proceedings of the Workshop on Software Engineering for Embedded Systems*, pages 40 – 46, 2003. Cited on page:13.

[37] IAN SOMMERVILLE AND PETE SAWYER. *Requirements Engineering: A Good Practice Guide.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997. Cited on page:8.

[38] JEFFREY J. P. TSAI, THOMAS WEIGERT, AND HUNG-CHIN JANG. A hybrid knowledge representation as a basis of requirement specification and specification analysis. *IEEE Trans. Softw. Eng.*, 18[12]:1076–1100, December 1992. Cited on page:16.

[39] KARL E. WIEGERS. *More About Software Requirements: Thorny Issues and Practical Advice.* Microsoft Press, Redmond, WA, USA, 2005. Cited on page:16.