Article can be accessed online at http://www.publishingindia.com

Formal Methods and its Importance in Minimizing Ambiguity in the Requirement Engineering Document Phase of SDLC

Shilpi Singh*, L.P. Saikia**

Abstract

The Requirements are the foundation for delivering quality software. And in today's business era different business organization needs commercial software and the software need to be delivered on time and within budget. And thus we need a complete, concise and accurate requirement engineering document. But most of the software which is delivered have errors, bugs and lack of functionality. And the reason for this is that the RE document is always written in natural language which is prone to ambiguities. And thus we need a mathematical method or a scientific method for writing RE documents. The formal methods in the past few years emerged as a most important or central issues in software engineering. And the precise specification, modeling and verification is identified in most of the engineering disciplines. And the formal methods provide us a tool to precisely describe a system correctly. The formal methods are a particular kind of mathematical techniques meant for specification, development and verification of software and hardware system. The primary reason of using mathematical formalism in RE text is the quality of the requirements and possibility of automation of some of the processes. In this paper we compare the different types of formal methods by analyzing there merits and demerits in the background of the different parameters like concurrency control, object oriented concepts and requirement engineering perspective

Keywords: Formal Methods, Lexical Ambiguity, Syntax Ambiguity, Semantic Ambiguity, Requirement Engineering Document (RE Text), Software Crisis

Introduction

One of the important phases of software development life cycle (SDLC) is requirement engineering. Since it is very important to understand the exact requirement of any system before actually building it. But this task is the most critical part in the SDLC phase. The RE document is written in informal language i.e in natural language. And the requirements are identified from the stakeholders (clients and users) who have inadequate knowledge to specify the requirement unambiguously and thus may lead to erroneous RE text. A requirement-engineering document is unambiguous if and only if every requirement has only one interpretation. Since, the developers and the customers are from different background and that's why they tend to understand things differently without knowing it.

RE is mainly concerned with gathering, analyzing, specifying and validating users requirement that are mostly documented using natural language [1, 2]. The IEEE standards mentioned that the ambiguous nature of the natural language makes the RE document ambiguous and thus degrades the overall quality of the RE document and they are called as "MEYER'S SEVEN SINS " and are as follows: Noise, Silence, Over – specification Contradiction, Ambiguity, Wishful Thinking and Forward reference [3].

Ambiguity in Requirement Engineering Text

The majority of requirement documents are written in natural language. It means that the requirements are imprecise as precision is difficult to achieve by just using natural language as the main presentation means. Natural language is the most used representation for stating requirements in industry. But it is inherently ambiguous. The customers and software developers may disagree on interpretations, which leads to disastrous software failure. Ambiguous requirements are the serious problems in software development because often

* Research Scholar, Department of Computer Science and Engineering, Assam Downtown University, Panikheti, Assam, India. Email: shilpi24_singh@yahoo.co.in

** Professor, Assam Down Town University, Department of Computer Science and Engineering, Panikheti, Guwahati, Assam, India. Email: lp_saikia@yahoo.com

stakeholders are not aware that there is some ambiguity. Ambiguity means that one phrase can be interpreted in several ways. It is one of the main problems that occur in natural language text. Ambiguities if noticed require immediate clarification [4, 5].

The RE text consists of different categories of ambiguity that may vary from *lexical ambiguity* to *semantic ambiguity* and again semantic ambiguity to pragmatic ambiguity. The RE specific ambiguity is context dependent (*Pragmatic ambiguity*). And thus the background of software development is very important and domain knowledge may also play a key role Moreover, lots of factor are responsible for introducing ambiguity in the RE text and thus complete and precise requirement specification is very difficult to achieve [6]. In many software system development projects the documents available for requirement analysis are always written in natural language. And ambiguity is intrinsic to natural language There are many recommended solutions to the ambiguity problem. One of the solutions is to follow some mathematical model or methods that can specify the exact requirement of the system [7, 8].

The ambiguity in the Requirement Engineering can be studied mainly in two stages. By the help of formal methods techniques we can specify the exact requirement of the system. And then by using any text classification technique we can detect and identify the ambiguities (Lexical, Syntactic, Sematic and pragmatic)[9]. And also we can measure the level of ambiguity [10,11,12]. We can improve the level of ambiguity in any RE text by following the formal methods to specify requirements and then detecting and classifying the ambiguity in RE text using text classification technique.

Formal Methods

The formal methods in the past few years emerged as a most important or central issues in software engineering. And the precise specification, modeling and verification is identified in most of the engineering disciplines. And the formal methods provide us a tool to precisely describe a system correctly [13]. The formal methods are a particular kind of mathematical techniques meant for specification, development and verification of software and hardware system. The primary reason of using mathematical formalism in RE text is the quality of the requirements and possibility of automation of some of the processes. The ambiguity in RE text is the biggest problem in the requirement elicitation phase and the exact science can remove it. The formal methods are mathematical based language technique that can be applied to specify the requirements of any system or prepare the RE document of any software system. The formal methods helps to specify the requirement of any software and hardware system by providing feature abstraction that helps to unambiguous description of any system. The formal methods helps to write the RE text scientifically which are based on the concept of set theory and first order predicate logic. The formalizing of the requirements can also used in the automatic test case generation and formal proofs can replace many test cases. But it does not show the correctness of the entire system. The notations used in formal methods is more complex than in graphical models but since it uses the basic mathematical notations and thus is not that hard to learn but and does not require trained mathematicians [14]. A formal method is a mathematical method to specify a hardware and/or software system verify whether a specification is realizable, verifies whether an implementation satisfies its specification, proves property of a system without necessarily running the system. Its specification language provides the mathematical basis of a formal method. More precisely, a formal specification language consists of two sets syntactic domain and *semantic domain* and a relation *satisfaction relation* between them.

Syntactic Domains: The syntactic domain of a formal specification language consists of an alphabet of symbols and a set of formation rules to construct well - formed formulas from the alphabet. And these well - formed formulas are used to specify a system.

Semantic Domains: Formal techniques can have semantically completely different semantic domains. Abstract data type specification languages are used to specify algebras, theories, and programs. Programming languages are used to specify functions from input to output values.

Satisfaction Relation: Given a model of a system, it is important to determine whether an element of the semantic domain satisfies the specifications. This satisfaction was determined by using similarities known as semantic abstraction function. It maps the element of the semantic domain into equivalent classes.

Formal Specification Styles

The formal methods are usually classified into two broad categories:

1. Model Oriented Style
2. Property Oriented style

Model based techniques are a way to write specification. To understand the nature of the system, model based languages helps to define a mathematical model of the system. The mathematical state model by the help of Entity types, Relationships types, sets, sequences and functions, constructs the state model of ant software system. The model-based languages such as Vienna Development Method (VDM), Zed (Z) and B method are used in many organizations to specify the systems requirements. Mainly the Property Oriented Style is used for requirement specification of any software system. Thus we will discuss about the pros and cons of Property Oriented Style of formal method techniques. In the property oriented style the system's behavior is defined indirectly by stating it's property usually in the form of set of axioms that the system must satisfy [15]. It is alleged that the property oriented style is more suitable for requirement specification because:

a) It specifies the systems behavior not by what they say of the system but what they do say about the system (Abstraction).

b) It permits a large number of possible implementations.

c) They specifies a system by a conjunction of axioms and thus makes it easier to alter specifications at a later stage.

The Property Oriented Style can further be classified as:

i) **Axiomatic Specifications** - The Axiomatic Specification uses the first order logic to write the pre and post conditions to specify the operations of a system in the form of axioms. The precondition basically captures the conditions that must satisfy before an operation can be successfully invoked. And for this we need to identify or capture the range of input that satisfies the function and establish the conditions as predicates. And then specify a predicate defining the conditions, which must hold on the output of the functions it behaved properly. And then establish the changes made to the functions input parameters after execution of the function. And finally combine all the above pre and post conditions of the functions.

ii) **Algebraic Specifications -** In Algebraic Specification Technique an object class or type is specified in terms of relationship existing between operations defined on that type. It defines the system as heterogeneous algebra. It is a collection of different sets on which several operations are defined. Each set of symbols in heterogeneous algebra is called a sort of the algebra. Traditional algebras are called homogeneous algebra. And thus for defining we need to specify involved operations, then signatures and there domains and ranges [16].

Properties of Algebraic Specifications:

a) Completeness b) Finite Termination Property

c) Unique Termination property

They are mainly designed for the definition of abstract data types and interface. The notations for developing algebraic specification languages are LARCH, ASL and OBJ. Various formal methods are reported in the existing literature. Some methods are significantly used in requirements phase and at software testing phase. A detailed discussion on some of the methods are given as follows:

Model Based Language

Z Method: Pronounced as " Zed" it was developed in 1977 by Jean – Raymond Abrail, It is based on set theory and first order predicate logic. The Schemas are used to describe both static and dynamic aspects of the system. A Z schema consists of a name, the signature part and predicate part. The attributes and the respective entities are specified in the signature part. And the predicate part consists of possible values. It is used to specify the operations of the system.

Merits:

a) It provides strong base for system design and testing because of high support on abstraction.

b) Free tools support is available in the market.

c) It is used to specify the functional aspects of the system.

d) It represents both static and dynamic aspects.

e) It helps to eliminate the inaccuracy in understanding the requirements written in natural languages.

Demerits :

a) It cannot be used to generate source code directly.

b) Lack of graphical notation.

c) It does not support concurrency control.

d) It does not support all aspects of design.

e) It does not specify that how these functionality can be achieved.

VDM: It the formal language developed in Vienna in the mid 1970s. It is a collection of different methods for the modeling, specification and design of software system. It describes the software system as models that represents the input, output and different operations on the objects. The requirement specification is typically given as abstract model. The objects identifies only the important aspects for expressing the essential concepts of the operation of the desired software system and hides the unnecessary features or details.

Merits: a) Specifications written in VDM can be useful in generating the source code directly.

b) It is comprehensive and precise and thus easy to understand.

c) It emphasizes on the feature of concurrency control.

d) It has the facility of explicit exception handling.

e) It can be useful to specify functional aspects of the system.

Demerits: a) It does not support all aspects of designing of a system.

b) It cannot specify usability, reliability, safety and performance requirement.

c) The error list can be removed from the system and thus it is difficult to see which errors are new and which are already been fixed.

B Method: Jean – Raymond Abrail in 1985, developed it. It is based on abstract machine notation. It is mainly used for specifying, refining and implementing software [17].

Merits: a) It can be directly used to generate source code.

b) Free tool support is available in the market.

c) The system design is refined and tested efficiently.

Demerits : a) It does not provide support for concurrency control.

b) It is slightly more low level.

c) It is focused on refinement of code rather than just formal specification

d) No support for object-oriented concept is provided.

Property Oriented Style (Algebraic Specification):

LARCH: They support a two – tiered definitional style of specification. One language that is designed for a specific programming language. Larch Interface Language (LIL) and another language that is independent of any programming Larch Shared Language (LSL). The top-level specification is mainly written in first order logic with equality and induction. The tiered system gives LARCH great flexibility since each target programming language gives specially designed BISL [18].

Merits:a) It uses interfaces between programs in different language.

b) It supports the feature of abstraction.

c) It is able to cope with the implementation of different implementation language.

d) It allows the specifiers to make assertions about specifications to validate specification before implementing.

Demerits: a) Most of the Larch specification cannot be executed.

b) It emphasizes simplicity and clarity rather than executability.

c) It is not focused to the major development phase.

OBJ : It is an algebraic programming and specification language introduced by Joseph Goguen . Object – oriented specification languages are those which provides at least there basic object oriented features (ie object, classes and inheritance). It is based on algebra, first order logic, temporal logic and set theory etc [19,20].

Merits: ***a)*** The formal specification language becomes more precise, accurate and easy to use after considering object-oriented language.

Object oriented concepts bridges the gap between static and dynamic specifications.

Inheritance helps in reusability of specifications.

An object oriented formal method can be merged and integrated in the whole software development life cycle.

Demerits: The adoption of the object- oriented paradigm adds powerful abstraction and structuring mechanism but at the same time creates problems and the number of interpretations and contradictory terminology makes it difficult to integrate or even compare different approaches.

Benefits of Formal Methods

The formal methods can be used at any stage of SDLC and thus has the capability to improve the quality of RE text. Some of the important goals are:

a) The formal methods help to clearly specify the exact requirements given by the user, which are not identical to the stated requirements. And the formal methods help to achieve this since they are unambiguous.

b) The formal methods help to provide early detection of errors and thus lead to early elimination of design defects.

c) It helps the people who write the requirement specification to ask all possible questions that could otherwise be postponed until coding. And even it has the property of completeness that covers all aspects of the system.

d) It allows us to do the rigorous analysis by which we can determine important properties such as satisfaction of high-level requirements or correctness of a proposed design.

e) By the help of formal methods we can design or systematically derive effective test cases directly from the specifications. It is cost effective way to generate test cases.

f) The formal methods usually have well-founded mathematical basis and thus are precise and accurate. The informal specifications are useful for understanding and documentation but cannot serve as basis for verification.

g) The formal methods have well defines semantics and thus ambiguity can be automatically avoided.

h) The mathematical basis automates the analysis of specification and it is one of the major advantages of formal methods [21, 22].

i) It can also be executed to obtain immediate feedback on the features of the specified system.

Limitations of Formal Methods

It is clear that formal methods provide mathematically sound background for writing an unambiguous requirement for a complex system, which will help to develop system in a systematic way rather than in an ad hoc manner. However, it has some limitations or shortcomings, which is important to consider. The formal methods are difficult to learn and use.

a) The basic incompleteness results of first order logic suggest that it is impossible to check absolute correctness of system using theorem-proving techniques.

b) They are not able to handle complex problems. And sometimes even the simple problem can become complicated by the use of formal methods.

c) A formal description of the program should contain a description that a program is to work in coordination with hardware and under the specification of operating system in order to prove the correctness of the program.

d) The correctness proofs play an important role in the formal methods. And it is generally impossible to ensure about the correctness of specification as well as implementation [23,24,25].

Comparative Study

For comparing the different formal methods we identified some of the attributes based on well known practices having similar cases. On the basis of the study of the different formal methods techniques (both model and property oriented) we have done a comparative study. The main parameters or the attributes considered for this study are as follows:

a) Concurrency Control: This parameter allows several computations to execute simultaneously and potentially interact with each other.

b) Supporting Tools: It helps in automation of any process. It makes the steps easier and thus is highly recommendable.

c) Support for Abstraction: It only captures those aspects that are relevant to the current perspective.

d) Object Oriented Concepts: The object oriented concepts like classes and objects are helpful for making software system.

e) Requirement Phase Perspective: It is the backbone of any software to be developed. Without any proper requirement specification we cannot proceed for accurate and precise software system. And this is recommended by the research community to consider and follow this concept for an unambiguous requirement specification phase.

Attributes	*Z Method*	*VDM*	*B Method*	*OBJ*	*Larch*
CONCURRENCY CONTROL	×	✓	×	✓	×
SUPPORTING TOOLS	✓	✓	✓	✓	✓
OBJECT ORIENTED CONCEPTS	✓	✓	×	✓	✓
REQUIREMENT PHASE PERSPECTIVE	✓	✓	×	×	✓

A table is made for this comparison so that we can have a clear picture of different features and aspects of the formal method techniques especially in the requirement specification phase.

From the above analysis we can draw a conclusion that the B method and the OBJ are not good for specifying RE document but the Z method, VDM and LARCH will help in the proper documentation of RE document.

Conclusion

Formal methods in software engineering are mathematical techniques that are used in the design, implementation and testing of computer systems. The application of mathematical methods in the development and verification of software is very labor intensive, and thus expensive. Therefore, it is not feasible to check all the wanted properties of a complete computer program in detail. It is more cost effective to first determine what the crucial components of the software are. These parts can then be isolated and studied in detail by creating mathematical models of these sections and verifying them. And the major defects in the software arise due to poor requirement analysis. And the formal methods are only part of the solution to the problem related to requirement analysis since, they help in the concise and precise specification of RE document and thus minimizes the ambiguity in RE text. The requirement is the basic building block of any software system in which the entire software development depends. This paper mainly presented the comparative study of different formal methods based on the parameters selected so that the requirement engineers could select a particular formal method for requirement specification. It is just one step in improving the requirement elicitation phase of SDLC so that the ambiguity can be removed or minimized from the RE document. From the discussions above we can see the B method and the OBJ are not benefitting but the Z method, VDM and LARCH will help in the proper documentation of RE document. And thus these methods help us to minimize the ambiguity from the text and leads to better software development. The researchers have made significant progress in the area of formal methods but still lots of improvement is required.

References

[1] Hall, A. (1990). Seven myths of formal methods. *IEEE software*, 7(5), 11-19.

[2] Meyer, B. (1985). On formalism in specification. *IEEE software*, January, 2(1), 6-26.

[3] Boehm, B. W. (1981). *Software Engineering Economics*, Prentice Hall.

[4] Kamsties, E., Berry, D. M., & Paech, B. (2001). Detecting Ambiguity in requirement document using inspection. *In Workshops on Inspection in Software Engineering - Paris*, (pp. 68-80).

[5] Mich, L., & Garigaliano, R. (2001). *Ambiguity measure in Requirement Engineering*, (pp. 39-48).

[6] Kamsties, E., Knethen, A. V., Philipps, J., & Schatz, B. (2001). An empirical investigation of the defect detection capabilities of requirements specification languages. *In Proceedings of the Sixth CAiSE/IFIP8. International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, (pp. 125-136).

[7] Osborne, M., & Macnish, C. K. (2001). Processing natural language software requirement specification. *Proceedings of the International Conference on Requirement Engineering*, (pp. 229-236).

[8] Mich, L., & Garigaliano , R. (2000). *Ambiguity measure in requirement engineering*. (pp 39 -48).

[9] Kamsties, E., & Peach, B. (2000). Taming Ambiguity in Natural Language Requirements. *In Proceedings of 13th International Conference Software and Systems Engineering and their Applications*.

[10] Hussain, I. (2007). *Using text classification system to Automate Ambiguity Detection in SRS document.* Masters Thesis, Computer science and Software Engineering Department, Concordia University, Montreal Canada, August 2007.

[11] Ferrari, A., & Gneisi, S. (2012). Using collective intelligence to detect pragmatic ambiguities. *In Proceedings of RE Conferences*, (pp. 191-200).

[12] Fabbrini, F., Fusani, M., Gnesi, S., & Lami, G. (2001). The linguistic approach to the natural language requirements, quality: Benefits for the use of automatic tool. *In Proceedings of the twenty sixth annual IEEE computer society - NASA GFSC software engineering workshop*, (pp. 97-105).

[13] Pandey, S. K., & Batra, M. (2013). Formal methods in requirements phase of SDLC. *International Journal of Computer Applications*, May, 70(13), 7-14.

[14] Bijorner, D., & Cuellar, J. R. (1998). Software engineering Education: Roles of formal specification and design calculi. In Annals of Software Engineering, 6, 365-409.

[15] Wing, M., & Davies, J. (2014). *Formal Methods*, I and II LNCS, (pp. 1708-1709), Springer.

[16] Wikipedia. OBJ. Retrieved on March 18, 2013.

[17] Martin, D., Tom, K., Steve, L., & Ursula, M. (2013). *Lightweight Formal Methods for Computer Algebra Systems*, University of St Andrews, UK Retrieved on April 10, 2013.

[18] Guttag, J. V., Horning, J. J., Garland, S. J., Jones, K. D., Modet, A., & Wing, J. M. (1993). *Larch: Languages and Tools for Formal Specification*, January.

[19] Cheng, B. H. C., & Atlee, J. M. (2007). *Research Directions in Requirements Engineering*, IEEE, May, (pp 285-303).

[20] Spivey, J. M. (1998). *The Z Notation: A Reference Manual.* (2nded), New York: Prentice-Hall.

[21] Baldwin, T., Li, Y., & Alexe, B. (2013). *Automatic term ambiguity detection*. IBM Research, Proceedings of 51st Annual Meeting of Association for Computational Linguistic, August, (pp. 804-809).

[22] Wilson, W. (1997). Writing Effective Requirement Specification. *USAF Software Technology Conference*, Utah.

[23] Denver, C., Berry, D. M., & Kamsties, E. (2003). Higher quality requirements specifications through natural language patterns. *In Proceedings of the IEEE International Conference on Software-Science, Technology & Engineering. (SwSTE'03), IEEE Computer Society Press*, (pp. 80-89).

[24] Ryan, K. (1993). The role of natural language in requirement engineering. *In Proceedings of ISRE*, (pp. 240-242).

[25] Kiyavitskaya, N., Zeni, N., Mich, L., & Berry, D. M. (2008). *Requirements for tools for ambiguity identification and measurement in natural language requirement specifications*, (pp. 207-239). Springer- Verlag London Limited.