Comparative Study and Performance Evaluation of Formal Specification Language based on Z, B and VDM Tools

Shrishti Tamrakar, Anubhav Sharma

Abstract— Formal method provides specification, proving and verification of software. It targets the semantics rather than syntax of the source program and can be used to provide an unambiguous and consistent supplement to natural language. Most of the software is delivered with some bugs, lack of complete functionality and cost overrun. Formal methods are silver bullet for software industry for solving these problems. Model based formal methods are those in which the system is specified in terms of state models that is constructed using mathematical notions such as sets and sequences. There are popularly three model based formal methods- Z, B and VDM (Vienna Development method). Z notation is used at an abstract level based on set theory and first order predicate logic. B is slightly low-leveland more focuseson refinement to code rather that just formal specification. VDM uses a group of formal modeling languages, it offers syntax type checking and proof obligation generation capabilities. This paper compares and contrasts the strengths and weaknesses of the model oriented formal specification languages- Z, B and VDM based on various factors. It is found that VDM is better tool for formal specification than Z and B.

Index Terms—Formal methods, Software Engineering, Compiler Specification, verification, proof obligation, Z notation, B notation, VDM.

1 INTRODUCTION

SOFTWARE ENGINEERING can be defined as "The systematic approach to the event, operation, maintenance and retirement of the software system". The primary goal of software engineering is to boost the standard of software product. The analysis phase of software development involves project planning and software requirement definition. The software requirement specification is a technical specification of requirements for the software product. The goal of software requirement specification definition is to completely and consistently specify the technical requirements for the software product in a concise and unambiguous manner, using formal notations as appropriate. The software requirement specifications based on the system definition. The requirement specification will state that the 'what' of the software product without implying 'how'. For Software formal specification, formal methods are used which are mathematical-based techniques used for specification, proving and verification of software systems [1]. The process of formal verification means that applying these approaches to verify the properties making certain correctness of a system. Formal verification of software system targets the computer program wherever linguistics of the language provides precise aiming to the program analyzed. Formal specification can be used to provide an unambiguous and consistent supplement to natural language descriptions and can be rigorously validated and verified leading to the early detection of specification errors [2].

Various researches have been made in different fields by using Z [17], B [11] and VDM [16] formal methods. Almeida et.al [7] in 1992 transformed a semi formal specification to VDM. Descriptions of the requirements of a software system written in an unconstrained natural language are considered to be informal. Informal descriptions are known to have the potential to contain ambiguities, partial descriptions, inconsistencies, and incompleteness and poor ordering of requirements, Specifications written in VDM like language are considered formal. In between these two ends they recognized several techniques for semi-formal specifications. In this paper they proposed a technique for semi-formal specification.

Ledru [9] in 1993 developed a reactive system in a VDM framework. This paper studied detailed development of reactive systems, using an extension of VDM. The extension allows specification and proof of behavioral aspects to be expressed in the VDM framework. This is achieved by using traces of the input/output activities and introducing the notion of external entities whose behavior is described by a state machine. The major objective of this work is to improve understanding of the practical implications of the specification, design, and symbolic validation of machine-checked reactive systems.

Ponsard et.al [20] in 2006 analyzed formal requirement models to Formal Specifications in B. They analyzed that the development of critical systems requires a high assurance process from requirements to the running code. Formal methods, such as B, now provide industry-strength tools to develop abstract models refine them in more concrete models and finally turn them into code. A major remaining weakness in the development chain is the gap between textual or semi-formal requirements and formal models. In this paper, they explore how to cope with this problem using a goal-oriented approach to elaborate a pertinent model, including regulation modelling, and turn it into a high quality abstract formal specification.

Dantas [11] in 2009 presented verified compilation and the B method: A Proposal and a first appraisal. This paper investigates the application of the B method beyond the classical algorithmic level provided by the B0 sub-language, and presents refinements of B models at a level of precision equivalent to assembly language. It claim and justify that this extension provides a more reliable software development process as it bypasses two of the less trustable steps in the application of the B method : code synthesis and compilation. The results presented in this paper have a value as a proof of concept and may be used as a basis to establish an agenda for the development of an approach to build verifying compilers based on the B method.

Zafar et.al [3] in 2011 worked in Transformation of class diagrams into Formal Specification. He says that requirement analysis and design specification is a serious issue in software engineering because of semantics involved in the transformation of real world problems to computational models. Unified Modeling language (UML) has been accepted as a standard for design and development of object oriented systems. UML has a lack of notations for description of a complete functional system and its semantics is still semi-formal allowing ambiguities at desing level. Formal methods involve much mathematics. Therefore, a strong linkage of UML and formal methods is needed to overcome the above issues. In this paper, an integration of UML and Z notation is defined for class diagrams considering both the syntax and semantics at an abstract level of specification.

Buragga et.al [4] in 2011 analyzed formal parsing of CFG (Context-Free grammar) using Left most Derivations. Formal approaches are useful to verify the properties of software and hardware systems. Formal verification of a software system targets the source program where semantics of a language has more meanings than its syntax. Therefore, program verification does not give guarantee the generated executed source code is correct as described in the source program. This is because the compiler may lead to an incorrect target program due to bugs in the compiler itself. It means verification of compiler is more important than verification of a source program to be compiled. In this paper, context-free grammar is linked with Z notation to be useful in the verification of a part of compiler. Firstly they defined the grammar, then language derivation procedure is described using the left most derivations. Next, verification of a given language is described by recursive procedures. The ambiguity of a language is checked as a part of the parsing analysis. By reading all these literatures it is found that B and Z notations have been used mostly, but the same work can be done through VDM which will give better output and easy to be work with.

The paper is organized as follows: Section II provides the problem definition of formal specifications like in Software industry where the problem lies in specifying the software and pros and cons of formal specification. Section III presents the applications and descriptions of each formal method-Z, B and VDM. Section IV reports the result of the survey of all the three formal methods. Finally, Section V draws preliminary conclusions on this survey and an agenda for future research.

2 PROBLEM DEFINITION

In software development, there's a tangle that development price will increase by back track once bugs that square measure enclosed within the section of demand definition square measure found within the when phases. As a good technique to unravel the matter, there is a technique with a proper specification language for demand. A formal specification language can describe the functional requirements exactly with mathematical in and verifies the specification to executable program with stepwise refinement. At stepwise refinement step, it will be found bugs by proof of the specification. [3]

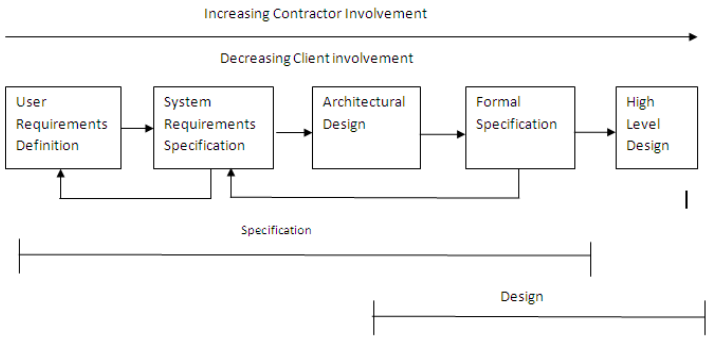

Fig.1 Formal Specification

Fig.1 depicts where the Formal Specification lies in the Software Development Life Cycle. It illustrates that between the User requirement Definition and the High level design there exists the Formal Specification. Its advantage is that it increases the contractor involvement rather than the client involvement.

Formal strategies haven't become thought software system development techniques as was once expected. Other computer code engineering techniques are made at increasing system quality. Hence, the necessity for formal ways has been reduced. Market changes are created time-to-time instead of computer code with an occasional error count the key issue. Formal ways don't scale back time to promote. The scope of formal ways is restricted. They're not similar temperament to specifying and analyzing user interfaces and user interaction. Formal strategies square measure arduous to proportion to massive systems. Formal specification involves additional effort within the early phases of software development. This reduces needs errors because it forces a close analysis of the need. Unity and inconsistencies may be discovered and resolved. Hence, savings may be created as quantity of work on owing to demand issues is reduced.

3 FORMAL SPECIFICATION TOOLS- Z, B AND VDM

1. *The Z notation* for specifying and planning package has evolved over the simplest a part of a decade, and it is currently doable to spot a regular set of notations that, though easy, capture the essential options of the strategy. This causes that junction rectifier to even this modest step towards standardization of Z. The first of those is that the growing trend towards pc helps within the writing and manipulation of Z specifications. Whereas the specifier's tools amounted to very little quite word-processing facilities, they had enough inherent

flexibility to make small differences in notation unimportant. But tools square measure currently being engineered that relies upon grammar analysis and to some extent on linguistics analysis, of specifications. For these tools – syntax checkers, structure editors, type checkers, and many more – to be useful and reliable, there must be agreement on the grammatical rules of the language they support. Nazir Ahmad Zafar and Fawaz Alsaade [2] has researched for "Syntax-tree Regular Expression Based DFA Formal Construction". They said that – Compiler's functionality is translation of computer program in source language to a machine code. Because of its size and complexity compiler construction is an advanced research area. Higher-level languages are usually complex which increases the level of abstraction. Due to all these reasons, design and construction of error-free compiler became a challenge for which verification of compiler must be done using one of the formal specification languages. Here in this application, Z notation has been used to formally specify a compiler. The sequence of labor done is as follows- 1st syntax tree is represented supported the increased regular expression. Then formal description of vital operators, checking null ability and computing initial and last positions of internal nodes of the tree is delineated. Then the transition diagram is represented from the follow positions and reborn into settled finite automata by shaping a relationship among syntax tree, transition diagram and DFA. The model analysis is provided exploitation Z/Eves toolset.

2. *The B-Method* is a mathematical method that belongs to the "model oriented" approach to software construction. The method is founded on set theory in a way which is made as solid as possible by reconstructing the original Ermelo set theory within the method itself. The method is based on a series of embedded notations: the logical notation, the basic set notation, the relational notation, the mathematical object notation, the generalised substitution notation, and, at the highest level, that of the Abstract Machine. The notation employed by the B-Method depends on associate degree extension to Dijkstra`s calculus. The extension enables specification of operations in terms of preconditions and post conditions, and permits object-oriented styles. Bartira Dantas [11] presented "verified compilation and B Method: A Proposal and a first Appraisal". In this paper B-method has been investigated on the far side the algorithmic level and additionally presents refinements of B models at level of preciseness reminiscent of programming language. This extension provides additional reliability towards computer code development by prying 2 steps: code synthesis and compilation. The result has been given within the kind of proof of idea and might be wont to build valedictory compilers primarily based in B methodology.

3. *The Vienna Development Method* is a mature formal method whose origins go back to the IBM Vienna Laboratory in the 1970s. It is a formal method for the description and development of computer systems. Its formal descriptions (or 'specifications') use mathematical notation to provide a precise statement of the intended function of a system. Such descriptions are built in terms of models of an underlying state with a collection of operations which are specified by pre- and post-conditions. VDM designs are guided by a number of proof obligations whose discharge establishes the correctness of design by either data rectification or operation decomposition. Thus it can be seen that VDM addresses the stages of development from specification through to code. From the wide variety of tools available it single out the Overture Automatic Proof System (APS) and the VDMTools for type checking, interpretation and code generation. Peter Gorm Larsen [10] presented "Recent industrial Applications of VDM in Japan". He analyzed that there is an industrial use of VDM in Japan since the acquisition of VDMTools by CSK systems. This acquisition followed a very successful application of VDM++ in the development of two subsystems of the Trade one back office system for securities trading. FeliCa Networks also applied VDM++ in the development of a new generation IC chip for use as an electronic purse which can be embedded in a cellular telephone.

4 RESULT AND DISCUSSION

The three formal methods discussed in section III – Z, B and VDM are the most powerful tool which can be used for analyzing the formal specification in respective notations. There does not exist any computer tool which may guarantee about complete correctness of a computer model. Therefore, even the specification is written using any of the formal languages it contain potential hazardous or errors. It means art of writing a formal specification never assures that the developed system is consistent, correct and complete. On the other hand, if the specification is checked and analyzed with the computer tool support it certainly increases the confidence over the system to be developed by identifying the potential errors, if exist, in syntax and semantics of the formal description.

Comparison of Z, B and VDM:

Comparison Factor	*Z*	*B*	*VDM*
Formal Method Style	Model-oriented	Model-oriented	Model-oriented
Mathematical basis	Set theory First order predicate calculus	Set theory First order predicate calculus	Set theory First order predicate calculus
Appearance Difference	Keyword oriented	Boxes or schemas	Keyword oriented
Structuring	Abstract machine notation	Schema calculus which allows various schemas to be combined to form new schemas	None
Specification of state change	None	Before: undecorated variables	Before: Hooked variables

		After: primed variables	After: unhooked variables
Identification of inputs and outputs	Input and output parameters are given by an operation header: Output ← Operation name (Inputs)	Inputs: variable names ending in "?" Outputs: variable names ending in "!"	No explicit way of specifying
Concurrency	No support for concurrency control	No support for concurrency control	Provide support for concurrency control using VDM++
Object oriented concept	Support object oriented concepts such as polymorphism, inheritance and encapsulation using object Z.	No support for object oriented concept	Support object oriented concepts such as polymorphism, inheritance and encapsulation using VDM++.
Tool support	Z Word Z/Eves Fastest	AtlierB ProB	SpecBox Overture VDM tools
Code generation	Software requirement specification cannot be automatically converted into computer source code.	Software requirement specification can be automatically converted into computer source code.	Software requirement specification can be automatically converted into computer source code.

Table.1 Comparison of Z, B and VDM

5 CONCLUSION AND FUTURE WORK

Though Z, B and VDM are model based formal specification languages used for specifying user's requirements in mathematical language that can be proved, verified and tested unambiguously. While the journey of all the three languages starts at the requirements specification phase of the software development life cycle (SDLC) model, but their path divides after this phase. Z works on high level abstraction of a system and provides a strong base for system designing and then testing it. However, B models the system in an abstract machine notation that can be used further to design system, generate its code and then refine and test the same. VDM is used to prove the equivalence of programming language concepts. They all do not differ radically from one another, but at some factors they differ a lot.

By seeing the different characteristics of all the three methods, it can be concluded that as Z and B has already been applied to compiler's specification but still remain a challenge for error-free compiler. If this application is to be put through VDM, then this can be helpful for providing error-free compiler better than the previous work done.

REFERENCES

[1] Dr.Arvinder Kaur, Ms Samridhi Gulati, and Ms.Sarita Singh, "Analysis of three formal methods-Z, B and VDM" , ISSN:2278-0181vol. 1, June 2012.
[2] Nazir Ahmad Zafar and Fawaz Alsaade, "Syntax-tree regular expression based DFA formal construction", IIM,2012,4,138-146.
[3] Nazir Ahmad Zafar and Fahad Alhumaidan, "Transformation of Class Diagrams into Formal specification", IJCSNS,Vol.11 No.5,May 2011
[4] Khalid A.Buragga and Nazir Ahmad Zafar, "Formal Parsing Analysis of context-free grammer using left most derivations", IARIA, 2011.
[5] Nazir Ahmad Zafar, "LR(K) parser Construction using Bottom-up formal analysis", JSEA, 2012, 5, 21-28.
[6] Nazir Ahmad Zafar, "Automatic Construction of formal Syntax tree based on regular expressions", 2012.
[7] Juliette D'Almeida, R.Achuthan, T.radahkrishnan,V.S.Alagar, "Transformation of a semi-formal specification to VDM", IEEE, 1992.
[8] Peter Gorm Larsen, kenneth Lausdahl, Nick Battle, "Combinatorial Testing for VDM", IEEE, 2010.
[9] Y.Ledru, "Developing reactive systems in a VDM framework", Elsevier, 1993, 51-71.
[10] Peter Gorm Larsen, "Recent Industrial Applications of VDM in Japan".
[11] Bartira Dantas, "Verified Compilation and the B method: A Proposal and a first Appraisal", Elsevier, 2009,79-96.
[12] Daniel Plagge and Michael Leuschel, "Validation B, Z and TLA+ using ProB and Kodkod", Technical Report, March 2012, Rev: 8536.
[13] Michael Leuschel and David Schneiderm, "Towards B as a high-level constraint Modelling Language Solving the jobs puzzle challenge".
[14] John Witulski and Michael Leuschel, "Checking computations of formal method tools - A secondary tool chain for ProB".
[15] Carlos A.L.Nunes and Ana C.R.Pava, "Automatic Generation of GUI from VDM++ specification", ICSEA 2011,pp.399-404, ISSN:2308-4235, Barcelona, Spain, October 23,2011 to October 29,2011.
[16] Juan C.Bicarregui, JohnS.Fitz Gerald, Peter A.Lindsay, Richard Moore, Brian Ritchie, "Proof in VDM : A Practitioner's Guide", 1st ed. Springer 1994. Available at http://overturetool.org/publications/books/proof-in-vdm/ProofinVDM.pdf
[17] J.M.Spivey, "The Z-notation-A Reference Manual", 2nd Edition, 1998. Available at http://spivey.oriel.ox.ac.uk/~mike/zrm/zrm.pdf
[18] Tony Hoare, "The verifying compiler- A grand challenge for computing research", Volume 50, Issue 1, Pages 63-69, ACM New York, USA, Jan 2003. Available at http://www.cs.ox.ac.uk/files/6187/Grand.pdf
[19] Andreas Muller, "VDM-Vienna Development Method", Bachelor thesis in "Formal methods in software Engineering", Johannes Kepler University Linz, April 20,2009. Available at https://www.yumpu.com/en/document/view/25785430/vdm-a-the-vienna-development-method-citeseerx
[20] Christophe Ponsard and Emmanuel Dieul, "From Requirements Models to Formal Specifications in B", available at http://ceur-ws.org/Vol-241/paper10.pdf
[21] D J Andrews and D C Ince, "Transformational data refinement and VDM", Elsevier 1995.