International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Comparative Analysis of Formal Specification Languages Z, VDM and B

Tulika Pandey[†] and Saurabh Srivastava[‡*]

[†]Department of Computer Science and Information Technology, SHIATAS, Allahabad, Uttar Pradesh, India
[‡]Department of Computer Science Engineering, SHIATS Allahabad, Uttar Pradesh, India

Accepted 20 June 2015, Available online 25 June 2015, **Vol.5, No.3 (June 2015)**

Abstract

This paper focuses on comparison on formal specification languages and chooses the appropriate one for a particular problem. Formal specification is a better way to identifying specification errors and describing specification in unambiguous ways. Formal specification is a specification written in a formal language where a formal language is either based on rigorous mathematical model or simply on standardized programming or specification language. Formal specification language expressed the specification in a language whose vocabulary syntax and semantic are properly defined. Formal specification language provides mathematical representation of the system. In this paper I will introduce three formal specification languages such as Z, VDM, and B and perform comparison among to them.

Keywords: *Formal Methods, Formal Specification, Formal Specification Languages, Schema.*

1. Introduction

This paper reports experience gained in use of formal specification languages in formal methods for complex and critical system development or decision making system. Formal methods are the mathematical approach supported by tools and techniques for verifying the essential properties of desired software system or hardware system. Formal methods are useful for checking the quality parameters such as correctness, completeness, and consistency and verification of the system requirements. Formal method is the integration of formal specification, formal proof and model checking. Formal specification is the first step in the formal development, it follow a series of steps involving verification and refinements of software system which leads to an eventual implementation. The primary role of the formal specification is to provide a precise and unambiguous description of the system for sub sequence steps. There are many formal specification languages are available but question is how they are different from one to another, this is a primary objective of this paper. In this paper I will compare three formal specification languages such as Z, VDM and B.

2. Formal Methods

Formal methods are the mathematical approaches supported by tools and techniques for verifying the essential properties of desired software system. Formal methods are useful for checking the quality parameters such as correctness, consistency, verification of the system requirements and provide code verification. Formal methods are associated with three techniques such as formal specification, formal verification, and refinements

$$\text{Formal Methods} = \begin{cases} \text{Formal Specification} \\ + \\ \text{Formal verification} \\ + \\ \text{Refinements} \end{cases}$$

Formal specification is a specification written in formal languages where a formal language is either based on rigorous mathematical models or simply on a standardized programming languages or specification languages. In this I will introduce three specification languages such as Z, VDM, and B and provide comparison among them according to different points of view. Formal verification is a process to prove or disprove the correctness of a system with respect to the formal specification or property. Refinement is a integral part of developing, checking, and verifying the specification.

3. Formal Specification Languages

The primary idea behind a formal method is that there are benefits in writing a precise specification of a system, and formal methods use a formal or mathematical syntax. This syntax is usually textual but can be graphical. A precise specification of a system can be used in a number of ways. First, it can be used as the process by which a proper understanding of the system

*Corresponding author: **Saurabh Srivastava**

can be articulated, thereby revealing errors or aspects of incompleteness. The specification can also be analyzed or it can be verified correct against properties of interest. For this purpose, a variety of different formal specification languages exist, with their proper tool support. A formal specification language is a specification language in computer science used during systems analysis, requirements analysis and system design to describe a system at much higher level than a programming, which is used to produce the executable code for a system. Specification languages are not directly executable. They are meant to describe the what, not the how. Indeed it is considered as an error if requirements are cluttered with unnecessary implementation details. Formal specification language provides mathematical representation of the system. Formal specification language expressed the specification in a language whose vocabulary syntax and semantic are properly defined. In this I will introduce three specification languages such as Z, VDM and B provide comparison among them according to various points of views whose description are present with the comparison results.

3.1 Formal Specification Language Z

Z is a constructive model-based specification language which was first suggested by Abrial and later developed at the University of Oxford and accepted as BSI standard in 1981. Z notation is based on the set theory and first order predicate logic. Z is popular especially in developing critical systems where the reduction of errors and quality of software is extremely important. It has undergone international standardization under ISO/IEC JTC1/SC22. Z provides a construct, called a schema, to describe a specification‘s state and operations. A specification in Z is presented as collection of schemas which can be combined and used in other schemas. Schema is a diagrammatic notation for displaying the predicates that are used in defining operations and invariants. The main building blocks of Z notation are basic type definition, axiomatic definition and schema definition. A basic types definition introduces one or more types which are used to declare different variables used in Z specification. An axiomatic definition is being used to describe one or more global variables and it optionally specifies a constraint on their values. In order to model an operation of any system, schema is being in the Z-Notation. A Z schema consists of a declaration and an operational list of predicates.

3.1.1 Terminology Used In Z Notation

Data Invariant: A data invariant is a condition that is true throughout the execution of the system.

State: In Z specification, the state is represented by the system‘s stored data.

Operation: Operation is an action that takes place within a system and read or writes data.
Conditions: Three types of conditions are associated with operation:

Invariant: An invariant define what is guaranteed not to change.

Precondition: A precondition defines the circumstances in which a particular operation is valid.

Postcondition: A Postcondition of an operation defines what is guaranteed to be true upon completion of an operation. This is defined by its effect on data.

3.1.2 Tool Support for Specification Language Z

Various tools for formatting, type checking and aiding proofs in Z are available. CADiZ is a UNIX based suite of tools for checking and typesetting Z specification. Z type checker (ZTC) and fuzz tool also support Z notation and type checking of Z specification. There is another tool named Z/EVES is also able to read entire files of specifications that have been previously prepared using LATEX markup. RoZ (Rosette) automatically generates the Z schemas corresponding to a UML class diagram.

3.2 Formal Specification Language B

B was developed by Jean-Raymond Abrial, also took part in the creation during the 1980s. B notation is closely related to Z and Vienna Development Method (VDM). B method has a strong decomposition mechanism. The primary aim of decomposition in B is to obtain a decomposition of proof. Formal verification of proof obligations ensures that a specification is consistent throughout its refinements. Like Z, B method is also based on first order predicate logic and set theory. The basic building block of B language is Abstract Machine. Gurevich initially introduce abstract state machine. An Abstract state machine specification is a program executed on an abstract machine. The program comes in the form of rules. Rules are nested if-then-else clause with set of function updates in their body. Based on those rules, an abstract machine performs state transitions with algebras as states. Firing a set of rules in one step performs a state transition. An abstract machine is a component that defines different clause such as SETS, CONSTANTS, PROPERTIES, VARIABLES, INVARIANT, INITIALIZATION and OPERATIONS but the order is not fixed. Microsoft research provides an executable version of Abstract state machine, which execute abstract state machine under .NET platform, i.e. AsmL.

3.2.1 Terminology Used In B Notation

Sets: The SETS Clause represents the list of deferred sets used in the machine.

Constants: Constants describe the type and properties of formal scalar parameters.

Properties: Properties clause shows the type and properties of machine constants.

Variables: Variables represent a list of abstract and concrete variables used in machine.

Invariants: Invariants also describes the type and properties of variables.

Initialization: Initialization clause is used to initialize the variables.

Operations: Operation clause list and define some specific operation.

3.2.2 Tool Support for Specification Language B

Two main commercial tools which support B language i.e. Atelier-B and B-ToolKit are used by researchers and developers. For methods B, there is a model checker tool, known as ProB, developed at the University of Southampton. The model checker ProB, includes as animator, which is amenable to validate the simulated behavior of specification. UML-B is a tool that translates UML class diagram and UML state chart diagram into B notation. But this tool work under certain conditions. Atelier-B proposes a set of commands allowing:

1. Syntax and Type checking of components.
2. Automatic generation of proof obligation.
3. Automatic demonstration of proof obligations.
4. Translatable language checking.
5. Translating into one of the following programming languages(C, C++, ADA, and HIA).

3.3 Formal Specification Language VDM

The Vienna Development Method (VDM) is a formal specification language which is model based language. VDM was initially developed for the formal description of PL/I at the IBM laboratory in Vienna. The VDM method considers the verification of step-wise refinement in the systems development process, i.e. data refinement and operation decomposition. VDM specifications are based on logic assertions of abstract states (mathematic abstraction and interface specification). In contrast to Z, VDM uses keywords in order to distinguish the roles of different components while these structures are not explicit in Z. As with Z specifications, VDM specifications are usually not machine executable. VDM supports the specification process by a mental execution with paper and pencil. However, proof assistance tools and tools for executing subsets are available.

3.4 Formal Specification Language VDM++

VDM++ is based on VDM-SL which is a formal specification language. VDM-SL is a model-based specification language whereas VDM++ is an object oriented extension of VDM-SL. [8]. In VDM++ the system is represented as a set of classes. Each class describes the state of the system. VDM++ classes have a powerful set of constructs such as constants, variables, operations and functions. More over VDM++ gives an explicit way to deal with the concurrency. To specify the real world active objects threads are used in VDM++. As distributed real-time systems are mostly concurrent systems, VDM ++ provides a full support to specify the distributed systems. The semantic and Syntax of VDM++ is easily understandable and close to a high-level programming language so it is very easy to adopt VDM++. VDM++ is a object oriented approach that provide concurrency support and synchronization. It is used for real time system and control system.

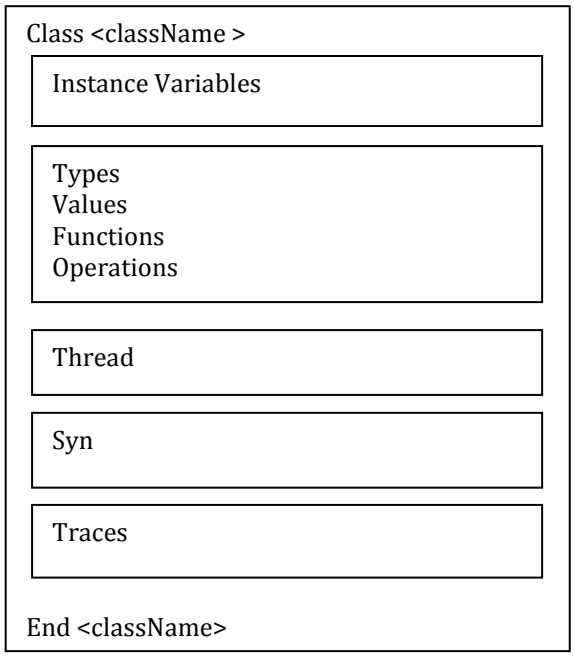

Fig. 1 VDM++ Class Outline

3.3.1 Tool support for VDM and VDM++

The tool support for VDM is very strong. A whole set of tools are available for VDM. The VDM tool kit provides following features

1. Syntax checker
2. Debugger
3. Document generator
4. Code generator for C++ and java
5. Interpreter
6. VDM++ to C++ code generator
7. Type Checker
8. Integrity Examiner
9. UML Link

4. Comparison and Results

4.1 Comparison Based on Specification Structure

Z specification structure can be defined by using schema. It has two parts first signature part identifiers are define or static component and second part is predicate part which has dynamic components. In the signature part contain list of declares the variable and predicate part contains list of the predicate. B specification structure using B machine element is shown below. VDM specification structure is not fixed. In VDM define sets, invariants functions, support constant values and some other. In VDM languages sets are finite because they contain only a finite number of elements. VDM language supports only two types of function first order and higher order.

Table 1 Comparison Based on Specification Structure

S. No.	Name of Specification Language	Notation
1.	Specification Language Z	*Schema Name* *Signature Part* *Predicate Part*
2.	Specification Language B	MachineName Sets Variables Constants Initialisation Invariants Operations
3.	Specification Language VDM	Type Values State End Function Opeation

4.2 Comparison Based On Specification Style

Property Oriented Specification defines the behavior of system indirectly by a set of properties in the form of axioms that the system must satisfy. Model Oriented Specification defines the behavioral of system directly by constructing a model of system. Often the level of abstraction determines whether a specification can be called property oriented or model oriented. Process Oriented Specification can be defined as let us consider a concurrent network of communicating components behavior and since most of the language describes system in term of process- oriented. All types of languages which belong to this category describe system in term of process. Sequential Oriented Specification defines input output behavior of system in sequential manner. That is output of first stage referred as input for next coming stage.

Table 2 Comparison Based on Specification Style

S. No	Name of Specification Language	Specification Style
1.	Specification Language Z	Sequential Oriented, Property Oriented and Model Oriented
2.	Specification Language B	Model Oriented
3.	Specification Language VDM	Process Oriented and Model Oriented

4.3 Comparison Based on Paradigm and Formal

Table 3 Comparison Based on Paradigm and Formality

S. No	Name of Specification Language	Formality and Paradigm
1.	Specification Language Z	Formal and State
2.	Specification Language B	Formal and State
3.	Specification Language VDM	Formal and State

4.4 Comparison Based on Concurrency

Concurrency is the property that allows many computations run simultaneously and potentially interact with each other's. Formal Specification Z, B and VDM do not support concurrency. In Object Oriented VDM approach provide Concurrency support.

Table 4 Comparison Based on Concurrency

S. No.	Name of Specification Language	Support Concurrency
1.	Specification Language Z	No
2.	Specification Language B	No
3.	Specification Language VDM	No,
4.	Specification Language VDM++	Yes

4.5 Comparison Based on Object Oriented Support

Z notation has features it break large specifications into smaller components. In object oriented approach system is distributed into objects each of which has its own sets of operations. In this way Z specification does not support object oriented concept but Object-Z is formal specification language which provide object oriented support hence make specification easier. Object oriented VDM support two types of module and inheritance mechanism, which are class modules and type modules. In class module define objects having their internal states. Type modules are those modules which specify objects with no states. Object oriented VDM inheritance mechanism are incremental inheritance and sub typing inheritance.

Table 5 Comparison Based on Object Oriented Support

S. No	Name of Specification Languages	Description of Object Oriented Support
1.	Specification Language Z	Support Object Oriented Concept using Object Z
2.	Specification Language B	Not Support Using Object Oriented Concept
3.	Specification Language VDM	Not supported to Object Oriented concept
4.	Specification Language VDM++	It Support Object Oriented concept

4.6 Comparison Based on Domain

This parameter states the specific domain of the language i.e. reactive systems, safety critical systems etc.

Table 6 Comparison Based of Domain

S.No.	Name of Specification Language	Domain Support
1.	Specification Language Z	Not Specific
2.	Specification Language B	Event B use in reactive and distributed system
3.	Specification Language VDM	VDM++ Real-Time systems, Control Systems

4.7 Comparisons Based on Timing Parameters

Timing Parameter**:** The timing constraints of real-time systems should be specified in the approved manner in order to ensure correct behaviour of real-time systems.

Discrete/ Continuous Time: Refers to whether the language handles discrete time or continuous time.

Input Time: This parameter evaluates the language on the basis of capability to handle input time. Real-time systems continuously interact with their environment and the input time is given by the environment.

Output Time: As input time is given by the environment, output time is purely specified by the system behaviour.

Table 7 Comparison Based on Timing Parameter

S.No	Name of Specification Language	Discrete/ Continuo-us Time	Input Time	Output Time
1.	Specification Language Z	Discrete	Yes	No
2.	Specification Language VDM	Discrete	Yes	Yes

4.8 Comparison Based on Executable

Executable specifications allow using specifications as a prototype hence executable specifications save the time and significantly reducing the cost.

Table 8 Comparison Based on Execution

S.No	Name of Specification Language	Executable
1.	Specification Language Z	No
2.	Specification Language B	Yes
3.	Specification Language VDM	No
4.	Specification Language VDM++	Yes

4.9 Uses in Industrial Application

In many industrial projects Z, B and VDM were used. Z used in Storm Surge Barrier Control System for formalization of design specification and use of Mondex Smart Card. METEOR Project has convinced Matra Transport International using B formal Specification. B method use for Metro Line 14 in Paris and Roissy Charles de Gaulle airport shuttle. VDM use in TradeOne System aim is to decrease the operating costs in trading securities. FeliCa Networks Project also use VDM++ which use to generate IC chip which is embedded in cellular phone.

4.10 Comparison Results

The objective of this work is to provide comparison of few formal specification languages that are considered as model based. Formal specification language are different from programming languages because the syntax and semantics of the specification language are more abstract the syntax and semantics of programming languages. Formal modeling provide constructs to write specifications of programming system, while programming languages provide constructs to write program. As the literature says, no single method can be truly applicable for all type of problems. Some specification languages such as Z, B and VDM are used for sequential system whereas other methods such as CSP and Petri Nets are used for parallel systems.

Formal specification language Z is more powerful approach that provides a precise specification but it is intractable. Formal specification language B is based on Z, but they are having some extended features. The primary aim of decomposition in B is to obtain decomposition of proof. B method is very useful for executable code generation that can also be used as an abstract specification language similar to Z, it ensure refinements steps and proofs, that the code satisfy its specification.

Conclusion

Formal Specification Languages are languages that are used to express the formal specification in a language whose vocabulary; syntax and semantics are formally

defined. This need for a formal definition means that the specification languages must be based on mathematical concepts whose properties are well understood. The branch of mathematics that is discrete mathematics is used and the mathematical concepts are drawn from set theory, logic and algebra.

Formal specification languages Z, B and VDM are model based languages. These languages support some parameters which are discussed above. On the basis of these parameters Vienna development method (VDM) language satisfy more parameters and it is the best language forms other two. Languages popularity also shows that VDM is more popular in the industry and projects. The use of Formal specification language in software development process do not force you to invest a significant cost or time overhead across the entire development, rather it is helpful for understanding the system being developed and preventing error from being propagated through the early stage of lifecycle to the later one. Many software engineering researchers proposed if we use formal specification languages in development by using formal methods that shows it was the best way to improve software quality. On the basis of discussion difference in syntax and structure, Z and VDM do not differ radically from one another. They are similar in their foundations and goals, and both allow the specifier to state requirements precisely and refine these specifications into designs correctly.

Use of the formal specification languages reduce the ambiguity and ensure the completeness and correctness of the software specification. A Model checker does not check programs, rather than it checks the properties of a model, which are high level descriptions of a system. In order to check whether the modeled system complies with the user requirements, it needs to verify and validate that particular model. Formal modeling is a task to convert a design document into a formal document, which is checked by model checking tools. In formal specification languages, tools such as, Z/EVES, Atelier B, VDM Tools, Alloy Analyzer etc. are used for sequential systems and PAT, CPN Tool, LOTOS tool, RSL tool, SPIN tool etc. are used for parallel systems respectively.

References

Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald (2009), Formal Methods: Practice and Experience. *ACM Computing Surveys,* Page 1-36.

Dobrica & Eilia Niemela (2002), *Survey on Software Architecture Analysis Method*s, IEEE Transaction of Software, Vol. 28, No.7

Ian Somerville (2009), *Introduction of Formal Specification*

M. Satish Kumar and Shivani Goel (2010), Specifying Safety and Critical Real Time in Z. *In International Conference on Computer and Communication Technology, IEEE, Pages 596-602*

Umesh Buller, Software Engineering and Formal Specification Lecture-6, IIT Bombay

R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luettgen, A. J. H. Simons, S. Vilkomir, M. R. Woodward, and H. Zedan, (20YY), Using Formal Specification to Support Testing, *ACM Journal*

Hassan Gomaa (2001), Software Design Methods for Concurrent and Real-Time Systems, *Addison-Wesley*

A. K. Sharma, Monika Singh (2013), Comparison of Formal Specification Languages Based upon Various Parameters, *IOSR –JCE,* Vol-11, page 37-39

H. Treharne, S. King, M. Henson, and S. Schneider (2005), *In ZB 2005: Formal Specification and Development in Z and B,* Lecture Notes in Computer Science, *Springer Berlin Heidelberg*.

Ashish Kumar Dwivedi (2014), Formalization and Model Checking of Software Architectural Style, *National Institute of Technology Rourkela*, Chapter 2, Page 9-14, Chapter- 4, page 29-38

Rabia Sammi, Iram Rubab, Muhammad Aasim Qureshi (2010), Formal Specification Languages for Real-Time System, *IEEE*, pp. 1642-1647

A. Laorakpong, and M. Saeki,(1993), Object-oriented formal specification development using VDM, Object Technologies for Advanced Software Lecture Notes in Computer Science, *Springer*, Page No. 529-543.

Kugamoorthy Gajananam and Prof. Helmut (2013), *National Institute of Informatics Technical Report*

Robert J. Allen (1997), *Formal Approach to Software Architecture*

Ian J. Hayes (1985) Applying Formal Specification to Software Development Industry, Vol SE-11, No. 2

Cooke D. and Gates, *Language for Specification Software*

R. Milner (1989), Communication and Concurrency, *Prentice Hall*

Kevin Lano, The B Language and Method: A guide to practical formal development, *Springer-Verlag*

J. R. Abrial, The specification language Z: Basic library," *Programming Research Group, Oxford Univ., Oxford, England,* Internal Rep., 1982

Jung Soo Kim and David Garlan (2010), Analyzing Architectural Styles, *Journal of Systems and Software*, 83(7), 1216-1235

Shouvik Dey and Swapan Bhattacharya (2010), *Formal Specification of Structural and Behavioural Patterns, JOT,* vol. 9, page 95-126

Robert J. Allen (1997), A Formal Approach to Software Architecture

Jones, C.B. (1986), Systematic software development using VDM, *Institute of Advanced Computer Science Prentice- Hall International*

Thomas McGibbon, An Analysis of Two Formal Methods: VDM and Z, *ITT Industries - Systems Division*

SPIVEY, J. M. (1989) The Z Notation: a reference manual' Prentice- Hall International.