Analysis of Three Formal Methods-Z, B and VDM

Dr.(Mrs.) Arvinder Kaur
Associate Professor,
University School of Information & Communication Technology,
Guru Gobind Singh Indraprastha University,
Dwarka, Sector-16C,
Delhi, India

Ms.Samridhi Gulati
M.Tech. Student
University School of Information & Communication Technology,
Guru Gobind Singh Indraprastha University,
Dwarka, Sector-16C,
Delhi, India

Ms. Sarita Singh
M.Tech. Student
University School of Information & Communication Technology,
Guru Gobind Singh Indraprastha University,
Dwarka, Sector-16C,
Delhi, India

Abstract

Formal methods provide a much needed solid software engineering foundation for the 'art' of programming computers. Formal specifications 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. Most of the software is delivered with some bugs, with lack of complete functionality and sometimes with cost overrun. Formal methods can be a silver bullet for software industry for solving these problems. This paper compares and contrasts the strengths and weaknesses of the model oriented formal specification languages such as Z, B and Vienna Development Method (VDM) basis of various factors.

1. Introduction

Formal methods are mathematically based techniques that can be applied throughout the development of a system to precisely describe a system and involve the use of refinement techniques and proof obligation at each stage to ensure the correctness, completeness and consistency of specification. The formal specification languages are based on set theory and first order predicate calculus, but this mathematical background was initially not fully formalized.[1]

In this paper we describe formal specification language and different formal specification styles in Section 2. In Section 3, Z, B and VDM are compared on the basis of factors such as characteristics, concurrency, object oriented concept, tool support, and code conversion. Conclusions are presented in Section 4.

2. Formal Specification Languages

This section describes an overview of formal specification languages.

The representation used in formal methods is called a formal specification language. The language is "formal" in the sense that it has a formal semantics and as a result can be used to express specifications in a clear and unambiguous manner. A formal specification language can be used to specify the task at hand in a clear and concise manner. As formal methods and formal specification language has sound mathematical basis, it provides the means of proving that specification is realizable, complete, consistent and unambiguous. Even the most complex systems can be modeled using relatively simple mathematical objects, such as sets, relations and functions [1].

A formal specification language is usually composed of three primary components or in mathematical term we can say that it consists of two sets syntax and semantics and a set of relation. [1]

The specific notation with which specification is represented is defined by syntactic domain or syntax. Formal techniques can have considerably different semantic domain. Semantics helps to define a "universe of objects" that will be used to describe the system. Set of relations defines the rules that indicate which objects properly satisfy the specification. Formal specification languages use mathematics as their basis. Most complex systems can be modelled using simple mathematical objects, such as sets, relations and

functions. A mathematical statement is unambiguous and precise, which provides a way to give convincing arguments to justify ones solutions, and allows proving that an implementation satisfies the mathematical specification [1].

Types of Formal Specification Styles

2.1 Model Based Languages

There are a number of different ways to write a precise specification. One approach is model based languages. In it the specification is expressed as a system state model. This state model is constructed using well understood mathematical entities such as sets, relations, sequences and functions. Operations of a system are specified by defining how they affect the state of the system model. Operations are also described by the predicates given in terms of pre and post conditions [2].

The most widely used notations for developing model based languages are Vienna Development Method (VDM) [4], Zed (Z) [3] and B [5].

2.2 Algebraic Specification

System behaviour is formally specified using a software engineering technique called, algebraic specification. These languages uses methods derived from abstract algebra or category theory to specify information systems. For the definition of abstract data types interface, algebraic approach was originally designed. Type operation is specified in order to define the type rather than the type representation. Ex LARCH, ASL, OBJ [6].

2.3Process Oriented

Concurrent systems are described using process oriented formal specification language. A specific implicit model for concurrency is the basis for these languages. In these languages processes are denoted and built up by expressions and elementary expressions, respectively, which describe particularly simple processes by operations which combine processes to yield new potentially more complex processes. Ex Communicating Sequential Processes (CSP) [7].

3. Comparison between Z, B and VDM

Z notation is a formal specification language that works at a high level of abstraction so that even complex behavior can be described precisely and concisely and it was originally proposed by Jean-Raymond Abrial in 1977 and was developed further at the Programming Research Group at Oxford University [8], whereas the B formal method is the method of software development based on B, a tool-supported formal method based around an abstract machine notation, used in the development of computer software and was originally developed by Jean-Raymond Abrial in France and the UK [9]. One of the longest-established Formal Methods for the development of computer-based systems is termed as Vienna Development Method (VDM).It was originated at IBM's Vienna Laboratory [10] in the 1970s.The Z, B and VDM are model based languages, which usually model a system by representing its state as a collection of state variables, their values and some operations that can change its state. All are based on set theory and mathematical logic. VDM was used in programming language description and compiler design. The main goal was to develop the language's fundamental features and to establish some formal semantics. Z notation is a strongly typed [11] mathematical, specification language. It is not an executable notation; it cannot be interpreted or compiled into a running program. Compared to Z, B is slightly more low level and more focused on refinement to code rather than just formal specification hence it is easier to correctly implement a specification written in B than one in Z.

3.1 Characteristics

The following table distinguishes Z, B and VDM based on different characteristics [12], [13], [14]

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	Key Word oriented (eg, PRE, THEN , END, INVARIANT)	Boxes or Schemas	Keyword oriented (e.g. pre-, post-, invariants)
Structuring	Abstract Machine Notation	Schema Calculus which allows various schemas to	None

		be combined to form new schemas	
Specification of State Changes	None	Before: undecorated variables After: primed variables	• Before: hooked 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

Table 1. Comparison on the basis of Characteristics

3.2 Concurrency

Concurrency is a property used in distributed system that enables software systems to be served in large-scale distributed systems. This property allows several computations to execute simultaneously, and potentially interact with each other.

Z	B	VDM
No support for concurrency control	No support for concurrency control	Provide support for concurrency control using VDM++

Table 2. Comparison on the basis of Concurrency

3.3 Object Oriented Concept

The object oriented concepts such as inheritance, polymorphism, and encapsulation are supported by some formal specification languages. Object oriented programming is an approach for developing software system based on the concepts of classes and objects. [15], [16]

Z	B	VDM
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++.

Table 3. Comparison on the basis of Object Oriented Concept

3.4 Tool Support

There are variety of robust commercially available tools that provide support for writing specification, proof obligation, refinement, syntax checking, type checking, editing, creating, proving theorems and many others. The table given below describes different types of tools and their features supported by Z, B and VDM. [17], [18], [19], [20], [21].

Z	B	VDM
Z Word Z/Eves Fastest	AtlierB ProB	SpecBox Overture VDM tools

Table 4. Comparison on the basis of Tool Support

3.5 Code Generation

Some formal specification languages support generation of computer source code from requirement specification.

Z	B	VDM

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 5. Comparison on the basis of Code Conversion

4. Conclusion

Though Z, B and VDM are model based formal specification languages used for specifying user's requirements in a mathematical language that can be proved, verified and tested unambiguously. While the journey of all 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 of 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 was used to prove the equivalence of programming language concepts as part of compiler correctness arguments. While specification written using B and VDM can be used to generate computer source code directly, Z notation doesn't provide such functionality. Free tool support is available for all three languages. Z, B 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 .Developing software with formal specification languages takes more time and hence money in comparison to the developing the same software without it but this access cost can be scarified in order to make software a reliable one and hence increase the trust of the customer in software industry.

5. REFERENCES

[1] R. Pressman, "Software Engineering- A Practitioner's Approach", McGraw Hill, 5th edition. 2000.
[2] D. Bjorner. Pinnacles of software engineering: 25 years of formal methods. In Annals of Software Engineering, vol 10, 2000, pp. 11–66.
[3] J. Davies and J. Woodcock, Using Z: Specification, Refinement, and Proof, In Prentice Hall, 1996.
[4] C. B. Jones, Systematic Software Development using VDM, In Prentice Hall, 1990.
[5] S. Schneider, B Method- An Introduction Palgrave, Cornerstones of Computing series, 2001.
[6] J. Guttag and J. J. Horning, The algebraic specification of abstract abstract data types, Acta Inform., vol. 10, pp. 27-52, 1978.
[7] C. A. R. Hoare, Communicating Sequential Processes, In Prentice Hall, NJ., 1985.
[8] J.R. Abrial, S. A. Schuman and B. Meyer: A Specification Language, in On the Construction of Programs, Cambridge University Press, eds. A. M. Macnaghten and R. M. McKeag, 1980 (describes early version of the language).
[9] J.R. Abrial. The B Book - Assigning Programs to Meanings. Cambridge University Press, 1996.
[10] D. Andrews. Report From The BSI Panel For The Standardisatio Of VDM (ist/5/50). In VDM '88 VDM | The Way Ahead. Springer Berlin/Heidelberg, 1988.
[11] J.M. Spivey, "The Z Notation,Reference Manual", 2nd edition, Prentice Hall International, 1992.
[12] J. Davies and J. Woodcock, Using Z: Specification, Refinement, and Proof, In Prentice Hall, 1996.
[13] C. B. Jones, Systematic Software Development using VDM, In Prentice Hall, 1990.
[14] S. Schneider, B Method- An Introduction Palgrave, Cornerstones of Computing series, 2001.
[15] B. Mahony and J. S. Dong, Timed Communicating Object Z, In IEEE Transactions on Software Engineering, vol. 26, No. 2, February 2000.
[16] Peter Gorm Larsen. VDM++ Tutorial At FM 06. Handout, 2006. http://fm06.mcmaster.ca/VDM++.
[17] The B-Toolkit". B-Core (UK) Limited. Archive.org. 2004. Retrieved February 22, 2012.
[18] M. Saaltink, "The Z/EVES System," in ZUM '97: The Z Formal Specification Notation, J. Bowen, M. Hinchey and D. Till, Eds., 1997, pp. 72–85.
[19] URL http://sourceforge.net/projects/zwordtools/.
[20] Steria, Aix-en-Provence, France. Atelier B, User and Reference Manuals, 1996.Available at http://www.atelierb.eu/index_en.html.
[21] CSK SYSTEMS CORPORATION. VDMTools User Manual (VDM++) ver.1.2, 2008. http://www.vdmtools.jp/en/.