Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

252

VDM '87
VDM – A Formal Method at Work

VDM-Europe Symposium 1987
Brussels, Belgium, March 23–26, 1987
Proceedings

Edited by D. Bjørner, C.B. Jones,
M. Mac an Airchinnigh and E.J. Neuhold

Springer-Verlag
Berlin Heidelberg New York London Paris Tokyo

Editors

Dines Bjørner
Department of Computing Science, Technical University of Denmark
DK-2800 Lyngby, Denmark

Cliff B. Jones
Department of Computer Science, University of Manchester
Manchester M13 9PL, United Kingdom

Mícheál Mac an Airchinnigh
Department of Computer Science, University of Dublin
Trinity College, Dublin, Ireland

Erich J. Neuhold
Gesellschaft für Mathematik und Datenverarbeitung mbH
Braunshardter Weg 11, D-6100 Darmstadt, FRG

CR Subject Classification (1987): D.2.2, D.3.1, F.3.2

ISBN 3-540-17654-3 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-17654-3 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.

Foreword

VDM-Europe is a technical advisory group to the *Commission of the European Communities* (CEC) in the general domain of *Information Technology*. As such, VDM-Europe is responsible to the CEC with respect to all issues concerning the specific formal software development method VDM[1]. In particular, it holds the brief to bring about more widespread awareness, use, development and standardisation of VDM in the *European Economic Community* (EEC).

The CEC recognises the important role that standards play in the harmonisation and development of the European software industry. Whereas the CEC does not itself devise such standards, it does use its good offices to promote such standards whenever and wherever they exist with the expressed intent to support the development of an open market within the EEC. A standardised use of VDM will give a strong competitive edge to the European software industry with respect to one formal software development method.

VDM 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's design approach is guided by a number of *proof obligations* whose discharge establishes the correctness of design by either data reification or operation decomposition.

Thus it can be seen that VDM is a complete development method which addresses all of the stages of development from specification through to code. It has now reached such a stage of maturity that (commercially available) support tools are reported on in this Symposium.

The origins of the IBM Laboratory in Vienna go back to a group which Heinz Zemanek brought from the *Technische Hochschule* (now the "Technical University of Vienna"). The group initially worked on hardware projects. A compiler for ALGOL 60 followed. The recognition that language definition was a crucial issue for the future safe application of computers was emphasized by IBM's creation of the PL/I language. The Vienna group built on ideas of Elgot, Landin and McCarthy to create an *operational semantics* approach capable of defining the whole of PL/I including its TASKING features which involved parallelism. These massive reports were known as the "Universal Language Document 3" and appeared in three more or less complete versions. The meta-language used was dubbed by outsiders the "Vienna Definition Language" or VDL. As well as such people as Kurt Bandat, Hans Bekič, Peter Lucas and Kurt Walk, one member of the current Programming

[1]The "Vienna Development Method"

Committee (EN) was involved in these descriptions and another (CBJ) in the attempt to use them as the basis for compiler design in 1968/70.

The attempts to use the VDL definitions in design were in one sense successful; but they also showed clearly how the operational semantics approach could complicate formal reasoning in an unnecessary way. The Scott/Strachey /Landin work on *denotational semantics* was taking shape in Oxford, Hans Bekič had long been pressing the Vienna group to adopt a more mathematical approach, and CJ had shown a "functional semantics" for ALGOL 60 in a Hursley Technical Report. The challenge, starting in late 1972, to design a compiler which translated the evolving ECMA/ANSI standard PL/I language into the order code of a completely novel machine presented the ideal opportunity to try out the denotational semantics approach.

Another member of the PC (DB) joined the Vienna group in 1973 and CBJ also returned. The project was fraught with difficulties and did not result in a finished compiler because of IBM's decision to abandon the machine architecture. But it did create VDM.

The formal description of PL/I in a denotational style is contained in a 1974 Technical Report which was authored by Hans Bekič, Dines Bjørner, Wolfgang Henhapl, Cliff Jones and Peter Lucas. The notation used became known as "Meta-IV" (both this awful pun and the name "VDM" are due to DB).

The diversion of the IBM group to handle more practical problems led to its dissolution. Among others to leave, Wolfgang Henhapl became a Professor in Darmstadt, Peter Lucas moved to IBM Research in the US, and DB took a visiting chair at Copenhagen and then a permanent one at the Technical University of Denmark. Of the key people only Hans Bekič remained pursuing — in his spare time — important research on parallelism until his untimely death in 1982 (see Springer LNCS 177).

Like other dispersions of scientists, this one did not kill the ideas but led to a larger community. The first step was to publish what had been done and DB/CJ edited Springer's LNCS 61 to this end. DB pursued the language description and compiler development work with Danish colleagues. This led to descriptions of both Ada (see Springer's LNCS 98) and CHILL and the first validated European compiler for the Ada language. CJ picked up the work he had been doing on formal development methods for non-compiler problems. Between them, DB and CJ have published three books with Prentice Hall International on VDM. There are also numerous papers tackling problems such as parallelism.

Peter Lucas has applied formal methods to application problems and Wolfgang Henhapl has worked on a support system (PSG) for VDM specifications.

In fulfilment of the responsibilities set out by its brief, VDM-Europe has identified five specific areas of endeavour:

1. **Technology Transfer** awareness and use of VDM shall be promoted by means of education and training.

2. **Application** strategies and tactics adopted, and experience in the use of VDM, shall be recorded and analysed.

3. **Support Environments** issues such as the portability and reusability of specifications written in the VDM meta-language shall be addressed. In particular, the development of tools which support VDM will be of special interest.

4. **Foundations** as with all formal methods, VDM's mathematical foundations must be sound and complete. By its very nature, which is essentially mathematical, VDM will continue to evolve. There are open areas of research which need to be addressed. There are explicit relationships with other branches of mathematics that need to be elaborated.

5. **Standardisation** whilst VDM is continuously evolving, stable concentric kernels must be established for standardisation.

To ensure a coherent programme which addresses each of the areas of endeavour of VDM-Europe and to bring state-of-the-art research and practice in VDM to a wider audience, the Programme Committee has adopted a symposium[2] format for this meeting. Of the originators[3], Dines Bjørner, Cliff Jones, and Peter Lucas will present key papers. The other papers have been solicited from the respective VDM experts in their field.

VDM is a formal software development method that is practised in the software industry. Such is its state of maturity that the Programme Committee are pleased to announce that a significant number of the speakers come from industry. There is a major upheaval taking place in the whole philosophy of software development. The coining of the phrase "software engineering" has forced a comparison with traditional engineering disciplines where an end-product results from a very specific engineering process that may be partitioned into distinct phases. The experience in the use of VDM in the software industry will be of major significance in fuelling the revolution that is currently taking place in software development methods.

In VDM, as in mathematics in general, one notes the continuous tension that exists between researchers and practitioners. On the one hand, new ground is being broken, frontiers are being extended. On the other hand, that which was once novel and revolutionary is now stable. The stability of a VDM kernel is essential for the construction of tools and the associated

[2]Interestingly, among its definitions, the Oxford English Dictionary gives the following: "symposium: 1. Drinking party, esp. of ancient Greeks with conversation etc. after banquet; 2. Philosophical or other friendly discussion; ...".

[3]The"ancient Greeks"?

VDM Support Environments. It is just such a kernel that is the basis of the standardisation process. However, life at the frontier looks very different. The VDM novice is confronted immediately by a diversity of notations, all of which purport to express specifications in VDM. Such diversity is healthy and may be ascribed to several historical factors:

- Printing Technology - originally being a "pencil and paper" method, the VDM meta-language notation was bounded only by the ingenuity of those who used it. However, VDM material in published form gave rise to notational variants dictated by the available printing technology.

- Application Domains - the uses to which VDM was put differed to such an extent that methodological as well as notational variants sprang into existence. The most obvious of these differences is between the use of VDM in the description of programming languages and in other applications.

- Pedagogy - without wishing to be dogmatic, one may identify two extreme variants of the VDM meta-language notation. There is the one which most resembles mathematical notation, which is succinct, expressive and terse. At the other extreme—and this is the one most likely to appeal to conventional software practitioners—there is a verbose, programming-like notation. Pedagogical concerns tend to force a particular form of notation that is most likely to be accepted and adopted by students of VDM who share a particular culture.

The papers in this Symposium reflect that reality. It is imperative to realise that, irrespective of the particular notational variant used in a given paper, the underlying formal development method is VDM. The sessions on Standardisation Issues and the Panel Discussion will clarify the position of the VDM kernel and that body of notation for which tools are and will be made available.

It has been noted above that VDM is but one formal software development method in use in the European software industry. However, its position is unique. A glance at the Symposium programme indicates the wide range of European users. It is now the only formal software development method for which there is a technological advisory group of the CEC and this Symposium will surely mark a new era in the development and use of VDM in Europe.

D. Bjørner
C.B. Jones
M. Mac an Airchinnigh
E. Neuhold

TABLE OF CONTENTS

VDM: Origins, Hopes, and Achievements
P. Lucas, Almaden, California, USA 1

VDM Experience

Experience Using VDM in STC
R.J. Crispin, Harlow, England 19

VDM in Three Generations of Ada* Formal Descriptions
J.S. Pedersen, Lyngby, Denmark 33

Experience with VDM in Norsk Data
U. Schmidt, R. Völler, Kiel, W. Germany 49

Use

Using VDM in an Object-Oriented Development Method for Ada* Sofware
Chr. Chedgey, S. Kearney, H.-J. Kugler, Dublin, Ireland 63

Development Methods

The Stepwise Development of Software Development Graphs: Meta-Programming VDM Developments
D. Bjørner, Lyngby, Denmark 77

Heap Storage Specification and Development
C. George, Harlow, England 97

VDM as a Specification Method for Telecommunications Software
Ths. Letschert, Nürnberg, W.Germany 106

VDM Environments

Support Environments for VDM
K.D. Jones, Manchester, England 110

Development and Application of a Meta IV Compiler
M. Haß, Kiel, W.Germany 118

From VDM to RAISE
S. Prehn, Lyngby, Denmark 141

Foundations I

Denotational Engineering or from Denotations to Syntax
A. Blikle, Warsaw, Poland 151

A Type Model for VDM
B.Q. Monahan, Cambridge, England 210

Specifications

A Formal Description of Object-Oriented Programming Using VDM
C. Minkowitz, P. Henderson, Stirling, Scotland 237

Foundations II

VDM Proof Obligations and their Justification
C.B. Jones, Manchester, England 260

Mathematical Structures and their Morphisms in Meta-IV
M. Mac an Airchinnigh, Dublin, Ireland 287

Standardisation Issues

Objectives of the British Standardisation of a Language to support the Vienna Development Method - The BSI VDM Specification Language Standardisation Panel - United Kingdom
D. Sen, Stevenage, England 321

Use of VDM within CCITT
P. Haff, Lyngby,
A. Olsen, Copenhagen, Denmark 324

A Case Study

A Formal Semantics for a DataFlow Machine - Using VDM
K.D. Jones, Manchester, England 331

Tutorial Papers

Introduction to the VDM Tutorial
M. Mac an Airchinnigh, Dublin, Ireland 356

Specification by Data Types
M. Mac an Airchinnigh, Dublin, Ireland 362

Data Reification and Program Decomposition
D. Andrews, Leicester, England 389

* Ada is a registered trademark of the U.S. Government (Ada Joint Program Office)