The return of VDM to Austria

Peter Gorm Larsen, IFAD, Denmark

May 18, 1998

Abstract

At the beginning of the seventies a new software development paradigm was invented at the IBM laboratories in Vienna [12]. To show respect for the origin of this technique its name was the Vienna Development Method (or VDM as an acronym). The main contribution of this work was the development of a specification language which was used to give a precise semantics to a complex programming language. The notation of VDM was standardised by ISO [13]. In addition it has been applied for making precise models of many different kinds of systems. During the last 5 to 10 years IFAD have been developing tools supporting this standard notation and an object-oriented version of it. In the last couple of years close collaboration with Institute for Software Technology[1] at TU Graz has emerged and we see this as a return of the VDM technology to Austria. It is like the ugly duckling returning home as a swan. In this paper we provide a short overview of the VDM technology and exemplify how we see the future collaboration with TU Graz. Finally we will express our vision for industrial use of VDM in Austria arising from this collaboration.

1 Introduction

The ugly duckling

The target domain for VDM was initially set narrowly on language definition and compiler design. One of the best known results of the early work was a formal definition of PL/I. Later on, the target domain was broadened to software and systems development in general. This was possible because the ideas were much more generally applicable than initially envisaged.

The method part of VDM for developing systems is to start with an abstract model of the functionality of the desired system. This is then followed by a series of precise models. Each model in this series is more concrete and closer to the implementation than the previous one. Every level is linked to the previous one by an argument of correctness. These steps would be considered as a refinement of the previous model. The underlying assumption is that one would be able to formally verify that the final implementation indeed satisfies the requirements stated in the first and most abstract model. In the early days of VDM rigorous proofs of certain compiler algorithms were actually developed in this way.

The returning swan

However, this idealistic development approach is difficult to introduce into industrial practice unless one has very well educated employees. The method part of VDM would not fit into traditional companies development processes without a revolutionary change. At IFAD we have therefore instead promoted a more evolutionary and light-weight approach [1]. We believe that the main benefit of VDM is to formulate the first abstract model of a system and validate whether the overall understanding makes sense. Introducing VDM gradually where one initially uses it simply for gaining a higher level of confidence in the understanding of the intended functionality is a more cost-effective way to introduce it in an industrial setting. Later on we envisage that the formal refinement and the proof of various properties may be introduced by companies which develop critical systems and have seen the benefit from doing the abstract modelling.

In order to use something like VDM industrially it is important to have powerful tools which can provide valuable insight in the VDM models being produced in an easy way. At IFAD we learned this in the beginning of the nineties and since then we have been involved with developing such tool support. Here a pragmatic ap-

[1] Ordinariat für Softwaretechnologie

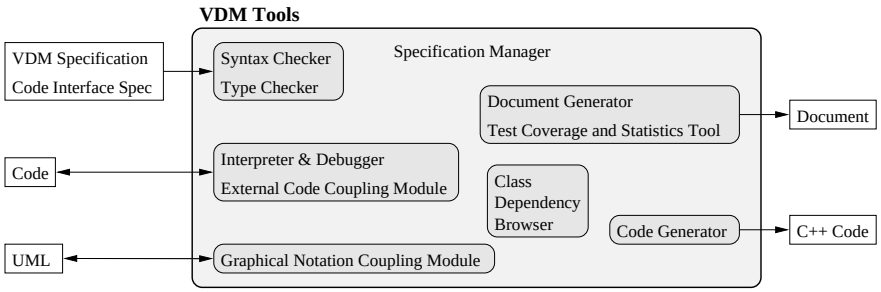

Figure 1: Overview of the IFAD VDM Tools.

proach focusing on validating VDM models using execution has been chosen. For most models formulated in VDM it is actually possible to execute them in a way similar to what developers use for their favourite programming languages. Thus, this is an easy way to gain valuable insight in the described functionality in an environment which is familiar to the traditional developers. In addition the platform for tools must fit with the traditional industrial environments. Nowadays this primarily means that the tools must run under Windows and support text processing systems such as Microsoft Word. At IFAD we have also moved our tools in this direction as described below.

We believe that it is due to our pragmatic approach that we have been able to get our VDM technology applied by industrial users. Some of these are confidential but for a number of them interesting papers have been written jointly with the industrial user. This includes work at British Aerospace [10, 7], work at Aerospatiale [11, 5], and work at Dassault Electronique [2]. In Austria students have started doing thesis projects in collaboration with Austrian industrial companies using the VDM technology. So far four such projects have been carried out and we expand on this below.

2 The VDM Tools: *Validated Design through Modelling*

An overview of the VDM Tools from IFAD is presented in Fig. 1 which illustrates how a number of facilities are integrated via a Specification Manager. Powerful analysis and consistency checking tools help the designer to achieve the highest confidence in system designs. The bi-directional coupling module for widespread graphical notations such as UML ensure the optimal support for complementary graphical and textual notations.

2.1 Executable Models: *Clarify Requirements and Find Defects*

A key facility of the VDM Tools is an interpreter and interactive debugger for executing and debugging high-level system models written in the VDM notations. This supports early validation and verification (V&V). For example, system requirements and hidden assumptions can be clarified by applying rapid prototyping techniques in order to validate system models, resulting in early trouble shooting and thereby reducing development costs and time-to-market. Furthermore, requirement properties can be documented explicitly by annotating VDM specifications with predicates which are verified automatically while executing test cases. Finally, the process of specification reviewing can be automated by this type of specification testing, thereby making it cheap to conduct and not subject to human errors nor limitations. Moreover, the reviewing process can be repeated at no cost which is particularly advantageous in situations with changing requirements.

Executable models are also essential from a technology transfer perspective. Software engineering practice is heavily based on testing and so this technique is well-known to engineers. The VDM Tools support the testing process by providing test coverage and statistics information at system model level and through a facility for executing models and external code together. This external code dynamic link facility supports rapid prototyping, for example, by building graphical front-ends for systems designed in VDM. Hence, it is possible to get early feedback on a model from a customer, manager

or colleague, who is not familiar with VDM.

2.2 Automatic Code Generation: *Reduce Costs and Time*

Automatic code generation helps to solve the consistency problem between specification and implementation and eliminates the human factor concerning the potential introduction of errors during implementation. Hence, automatic code generation can help to minimise development costs and time-to-market. Moreover, bug fixing and other maintenance can be performed at specification rather than implementation level (i.e. at a level corresponding to early rather than late development steps), yielding easier maintenance and lower costs.

The IFAD VDM Tools support automatic generation of C++ code from high-level models as an add-on feature. The tools generate fully executable code for 95% of all VDM constructs leaving facilities for including user defined code for non-executable parts of the specification.

2.3 Graphical Notations: *Master Complexity*

Graphical notations are widely used for object-oriented designs and embedded real-time systems. The Venus solution offers the complementary benefits of the textual notation VDM++ and the graphical notation UML for object-orientation through a bi-directional link to Rational Rose. Hence, Venus supports round trip engineering where an engineer can begin the development process by writing graphical models in UML, automatically translate these to VDM++, incrementally develop more concise models, test the models using the VDM++ interpreter, change class structure, merge with existing UML, make changes in UML, merge with existing VDM++, etc. The advantage of using VDM++ is that it allows more concise system specifications than with purely graphical notations and that these can be executed and debugged early in the development process.

2.4 Tool Descriptions

The tools presented in Fig. 1 are described in more detail below.

Specification Manager The Specification Manager maintains a project by keeping track of the status of modules. These status indications present an easy overview of the specification and they facilitate the continuation of work from one session to another. Furthermore, the Specification Manager can automatically update the specification with respect to syntax checking, type checking, code generation, etc. Finally, the user has a number of options, e.g. he can select his favourite editor.

Interpreter The VDM-SL/VDM++ interpreter supports all executable constructs in VDM-SL and VDM++. This range from simple value constructors like set comprehension and sequence enumeration to more advanced constructs like exception handling, lambda expressions, loose expressions and pattern matching. One of the benefits of executing specifications is that testing techniques can be used to assist validation of the specifications. In the development process small examples for parts of a specification can be executed to enhance the designer's knowledge of, and confidence in the specification. Furthermore, an executable specification can form a running prototype.

Debugger A source-level debugger is essential when working with executable specifications. The VDM debuggers supports the functionality found in debuggers for ordinary programming languages including: setting breakpoints, stepping, inspection of all variables defined in scope, and inspection of the call stack.

Type Checker The static semantic analyser is an advanced type checker and it supports most of the static semantics levels prescribed by the ISO Standard. It contains a powerful type inference mechanism which also shows proof obligations with respect to the type system.

Test Facility Test coverage information can be automatically recorded during the evaluation of a test-suite. The specifier can at any point view which parts of the specification are most frequently evaluated and which parts have not been covered at all. The test coverage information is displayed directly in the source-file, a Word or LaTeX document, in a comprehensive form which is easy to read.

Automatic Code Generator The IFAD VDM Tools support automatic generation of C++ code from a VDM-SL or VDM++ specification which helps to solve the consistency problem between specification and implementation. The code generator produces fully executable code for 95% of all VDM constructs leaving facilities for including user defined code for non-executable parts of the specification.

Dynamic Link Facility The IFAD VDM-SL Toolbox has an add-on feature which makes it possible to integrate external code into the execution of a specification. This can be used to integrate a formal model with components developed in a traditional way and provide graphical front-ends for a model.

Rose-VDM++ Link The Rose-VDM++ Link integrates UML and VDM++. Through translations the link provides a tight coupling of the IFAD VDM++ Toolbox and Rational Rose. Hence the link supports round trip engineering between UML and VDM++, where the graphical notation is used to provide the structural, diagrammatic overview of a model while the formal notation is used to provide the detailed functional behaviour of a model.

3 Collaboration with TU Graz

After the return of Professor Peter Lucas to Austria collaboration between IFAD and IST at TU Graz has emerged. At IST the students are taught VDM using the new VDM tutorial book [6] which includes an educational version of the tools mentioned above called Toolbox Lite.

The first visible result of the collaboration came from the work Brigitte Fröhlich did at a visit to IFAD. She developed the first version of the dynamic link feature mentioned above [9]. The subject of her PhD thesis is to what extent one can execute implicit definitions from VDM-SL [8]. This work can been closely followed and guided jointly by both IST and IFAD. The different conceptual ideas have been tried out using the VDM-SL Toolbox extending specifications already developed at IFAD. In case that the results are sufficiently encouraging the work here will be incorporated in the IFAD tools at a later stage.

The second visible result came from Bernhard Aichernig who carried out his MSc project at IFAD. The subject of this work was to automate the generation of proof obligations [3, 4] for VDM-SL. This is already scheduled to be incorporated in the IFAD VDM Tools. In his PhD work he is now looking at how one can automate a larger part of the testing efforts based on specifications formulated in VDM-SL or VDM++. Just like the work mentioned above joint guidance is given to his work including visits to Denmark to discuss and present alternative routes for the future work.

In August Oliver Oppitz will start his MSc project at IFAD. The subject of his work will be automatic code generation from the parallel part of VDM++ to Java. For the sequential part of VDM++ to Java code generator is currently being developed at IFAD. We believe that this will again be a successful visit with mutual benefits.

Industrial dissemination

In Austria students have started doing thesis projects in collaboration with Austrian industrial companies using the VDM technology. So far several projects have been carried out including a project with Siemens PSE Graz where several ambiguities and gaps have been discovered in the informal description of a protocol which has been formalised using VDM-SL. Another of these projects which deserves to be mentioned is one together with Ferk Informatik where VDM-SL is combined with structured analysis and entity-relationship diagrams. Currently the VDM technology is applied in a project in the safety-critical area of air control. The goal of this project is the specification and derivation of test-cases for the communication system of the Viennese company Frequentis. Furthermore, in an ongoing PhD project in cooperation with the Research Centre Seibersdorf a Guard Route Control System is formally redeveloped using VDM.

4 Concluding Remarks

Where VDM for a long number of years has been almost unknown in Austria we believe that the fruitful collaboration between IFAD and IST at TU Graz will mean the return of VDM to Aus-

tria. We believe that the TU Graz focus on VDM will spread into more and more industrial applications using the VDM technology. Here the students projects carried out on problems from the industrial companies play an important role.

We envisage that the collaboration will be extended where more students in Austria will be exploiting ideas for extending the VDM Tools using a new API (Application Programmers Interface) which has been developed for the Toolbox. In this way new features will be added and those which looks most promising will probably be further enhanced and maintained on a commercial basis by IFAD afterwards.

We really believe that this story with VDM returning to Austria is similar to that of the ugly duckling's return as a beautiful swan. In particular because IFAD comes from Odense in Denmark which is the home city of H.C. Andersen who wrote the fairy-tale.

References

[1] Sten Agerholm and Peter Gorm Larsen. A Lightweight Approach to Formal Methods. In *Submitted to FM-Trends*, Boppand, Germany, October 1998.

[2] Sten Agerholm, Pierre-Jean Lecoeur, and Etienne Reichert. Formal Specification and Validation at Work: A Case Study using VDM-SL. In *Proceedings of Second Workshop on Formal Methods in Software Practice*. ACM, Florida, March 1998.

[3] Bernhard Aichernig. A Proof Obligation Generator for the IFAD VDM-SL Toolbox. Master's thesis, Technical University Graz, Austria, March 1997.

[4] Bernhard K. Aichernig and Peter Gorm Larsen. A proof obligation generator for vdm-sl. In John Fitzgerald, Cliff B. Jones, and Peter Lucas, editors, *FME'97: Industrial Applications and Strengthened Foundations of Formal Methods (Proc. 4th Intl. Symposium of Formal Methods Europe, Graz, Austria, September 1997)*, volume 1313 of *Lecture Notes in Computer Science*, pages 338–357. Springer-Verlag, September 1997. ISBN 3-540-63533-5.

[5] Lionel Devauchelle, Peter Gorm Larsen, and Henrik Voss. Picgal: Practical use of formal specification to develop a complex critical system. In John Fitzgerald, Cliff B. Jones, and Peter Lucas, editors, *FME'97: Industrial Applications and Strengthened Foundations of Formal Methods (Proc. 4th Intl. Symposium of Formal Methods Europe, Graz, Austria, September 1997)*, volume 1313 of *Lecture Notes in Computer Science*, pages 221–236. Springer-Verlag, September 1997. ISBN 3-540-63533-5.

[6] John Fitzgerald and Peter Gorm Larsen. *Modelling Systems – Practical Tools and Techniques in Software Development*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 1998. ISBN 0-521-62348-0.

[7] John S. Fitzgerald. ESSI Project ConForm: Home Page. WWW at URL http://www.csr.ncl.ac.uk/projects/ConForm.html, 1994.

[8] Brigitte Fröhlich. *Towards Executability of Implicit Definitions*. PhD thesis, TU Graz, Institute of Software Technology, September 1998.

[9] Brigitte Fröhlich and Peter Gorm Larsen. Combining VDM-SL Specifications with C++ Code. In Marie-Claude Gaudel and Jim Woodcock, editors, *FME'96: Industrial Benefit and Advances in Formal Methods*, pages 179–194. Springer-Verlag, March 1996.

[10] Peter Gorm Larsen, John Fitzgerald, and Tom Brookes. Applying Formal Specification in Industry. *IEEE Software*, 13(3):48–56, May 1996.

[11] Peter Gorm Larsen Lionel Devauchelle and Henrik Voss. PICGAL: Lessons Learnt from a Practical Use of Formal Specification to Develop a High Reliability Software. In *DASIA'97*. ESA, May 1997.

[12] Peter Lucas. VDM: Origins, Hopes, and Achievements. In Airchinnigh Bjørner, Jones and Neuhold, editors, *VDM '87 VDM – A Formal Method at Work*, pages 1–18. VDM-Europe, Springer-Verlag LNCS 252, 1987.

[13] P. G. Larsen and B. S. Hansen and H. Brunn N. Plat and H. Toetenel and D. J. Andrews and J. Dawes and G. Parkin

and others. Information technology — Programming languages, their environments and system software interfaces — Vienna Development Method — Specification Language — Part 1: Base language, December 1996.