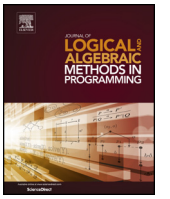
Journal of Logical and Algebraic Methods in Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



The evolution of VDM tools from the 1990s to 2015 and the influence of CAMILA

Peter Gorm Larsen [a,*], John Fitzgerald [b]

[a] Department of Engineering, Aarhus University, Denmark
[b] School of Computing Science, Newcastle University, Newcastle upon Tyne, UK

ARTICLE INFO

Article history:
Received 8 July 2015
Received in revised form 4 October 2015
Accepted 5 October 2015
Available online xxxx

Keywords:
VDM
CAMILA
Tool support
History
Industrial applications

ABSTRACT

The Vienna Development Method (VDM) is one of the most mature formal methods, with a history of cost-effective industrial deployment. One important route for this has been the development of robust tools supporting the construction of models, and their animation. We trace the history of this strand of work from the mid-1990s to 2015, taking as our starting point challenges for the industrial usage of formal methods set out by José Nuno Oliveira in 1997. We describe five generations of VDM tools: the IFAD VDM Toolbox, VDMTools, Overture, Crescendo and Symphony, emphasising the influence that the goal of industry usage has had on their features and architectures. We chart the move from a single-formalism tool focused on executable VDM specifications to a platform for multi-tool analysis of a wider range of models, and look forward to the growth of integrated multidisciplinary toolchains from the ongoing INTO-CPS project. We briefly compare the VDM tool story with the approaches taken by other formalisms that have been applied in industry.

1. Introduction

Decades of persistent research on applicable formal methods, often in the face of scepticism, are bearing fruit, and there are now many notable successes to report [1,2]. These are in large part due to the research and practitioner community better tuning formalisms to industrial needs, and developing the robust tools that are a *sine qua non* for effective application. It was not always clear that the goal of stronger tooling would be pursued, but the vision and contribution of scientists such as José Nuno Oliveira have been fundamental to achieving the positive state of formal methods today.

The Vienna Development Method (VDM) originated in the pioneering work of IBM's Vienna Laboratory on the challenges of defining programming language semantics and of designing trustworthy compilers. In his 1999 review of the scientific history of VDM, Jones remarked that "In spite of a number of efforts, projects to provide tool support have never been easy to justify" [3]. In the years since, the VDM community has set itself the goal of creating a formal method that is supported well by tools that deliver the capabilities needed for industry deployment. A measure of its success is to be found in significant applications like that of the FeliCa chip firmware, deployed in more than 259 million mobile telephones [4] using VDMTools [5].

We would argue that three principles have tacitly guided VDM tools work in the last 25 years. First, priorities for tooling have gone hand-in-hand with the needs of targeted industry application, leading to an emphasis on model construction

* Corresponding author.
E-mail addresses: pgl@eng.au.dk (P.G. Larsen), john.fitzgerald@newcastle.ac.uk (J. Fitzgerald).

http://dx.doi.org/10.1016/j.jlamp.2015.10.001
2352-2208/

Please cite this article in press as: P.G. Larsen, J. Fitzgerald, The evolution of VDM tools from the 1990s to 2015 and the influence of CAMILA, Journal of Logical and Algebraic Methods in Programming (2015), http://dx.doi.org/10.1016/j.jlamp.2015.10.001

and simulation. Second, tools robustness is as important as capability, with the consequence that a release discipline has been developed even for what is sometimes unfunded tool development. Third, tools should be integrated with, rather than supplant, established workflows; it is consequently necessary to understand workflows before deployment of tools.

This paper reviews the recent history of tools development for VDM, with its focus on support for model construction and analysis through simulation and industrial usage. In Section 2 we first elaborate concerns on the industrial take-up for formal methods, as articulated by Oliveira in 1997. These provided significant motivation for both the CAMILA framework and the emerging IFAD VDMTools (Section 3). We describe the influence of this work on the development first of VDM-Tools (Section 4) and then its open-source cousin Overture (Section 5), and discuss how that framework is being extended today to provide formal model-based tools for the design and analysis of increasingly demanding classes of product, notably Systems of Systems and Cyber-Physical Systems (Section 6). Finally, we briefly relate the VDM tool development to other formal methods (Section 7), discuss the extent to which we have succeeded or failed to meet Oliveira's challenges, and look to future directions (Section 8). Throughout, we aim to provide an extensive bibliography, particularly relating to the history of VDM's tool support.

2. Enabling industrial use of formal methods

In May 1997, Oliveira gave a presentation at the United Nations University International Institute for Software Technology [6], in which he listed his main concerns for industrial usage of formal methods as follows:

Simplicity: Industry will never absorb formal methods based on complex mathematical theories.
Compatibility: Formal methods cannot replace traditional methods altogether.
Tools: Formal methods should be supported by tools and environments.
Flexibility: Formal method tools should be easily portable across different machine platforms and be able to communicate with existing (traditional) tools.
Modularity: Common sense should be applied to software development; warning: informal modularity is worse than formal monolithic development.
Reusability: Requires a formal classification scheme; otherwise, repositories become full of things we will never find.
Back to "good engineering habits": In school physics we are taught a universal strategy for problem solving: understand the problem, build a mathematical model of your understanding of it, reason in this model, upgrade your model, if necessary, and calculate a solution. Why don't we do likewise in software development?

The authors of this paper were making similar observations to Oliveira at around the same time [7]. We were motivated by the experience of applying VDM at British Aerospace (Systems and Equipment), in an early comparative study of software development with and without formal models [8]. Our strongest focus was perhaps on Oliveira's *compatibility* and *tools* issues. With others we advocated a pragmatic "lightweight" approach to formal methods in which methods would remain fully formal but would be applied to subsystems and system features that merited the investment [9,10].[1] We also observed that industrial users rarely develop systems from scratch; instead they often build on existing solutions or use existing components from other projects. Thus, there is a clear need to lower the barriers to the use of formal techniques where legacy features exist.

Finding an appropriate balance between the effort spent on producing formal models and the value they bring either in the form of new insight or in the form of a product is paramount to the industrial application of formal methods [11]. In order to gain value from formal models it is important to supply efficient tool support that enables users rapidly to understand models, identify weaknesses and explore alternatives, and here the notion of executable models becomes important.[2]

A model-based formal method such as VDM has many features that appear familiar to software engineers with experience of imperative and functional programming. In working with engineers in industry we rarely found difficulties in understanding the elements of VDM as a modelling language; the most challenging skill to teach is the crucial one of abstraction [15]. Abstraction decisions should be governed by the *purpose* of the model [16], but models must remain rich enough to be *competent* in the sense that engineers should have confidence that the outcomes of model-based analysis will reflect the properties of the realisation. A focus on abstraction skills in formal methods education is both essential and, we argue is enhanced by the use of tools [17].

3. CAMILA/SETS and IFAD VDM-SL Toolbox developed in parallel

The CAMILA initiative at the University of Minho, led by Oliveira (the original project for its development was from 1990 to 1993) [18–21] developed a formal modelling notation inspired by functional programming and set theory. The focus here

[1] Bernhard Steffen once remarked to Fitzgerald that, although our approach might be called "lightweight", the specific gravity of VDM remained the same – in fact we had provided the machinery for lifting it!

[2] Many model-based formal methods now define executable subsets, but the value of executability was debated because of the risk of compromising abstraction [12–14].

Please cite this article in press as: P.G. Larsen, J. Fitzgerald, The evolution of VDM tools from the 1990s to 2015 and the influence of CAMILA, Journal of Logical and Algebraic Methods in Programming (2015), http://dx.doi.org/10.1016/j.jlamp.2015.10.001