Exploiting formality in software engineering

Juan C. Bicarregui

CLRC, Rutherford Appleton Laboratory
J.C.Bicarregui@rl.ac.uk,
http://www.itd.clrc.ac.uk/Person/J.C.Bicarregui

Abstract. There is undoubtedly a huge gap between the level of formality currently in use in mainstream software engineering and the "best practise" advocated by academics and practised by a small sector of industry involved in critical applications. This paper presents some results of recent research which are building bridges between these two approaches: on the one hand, developing formal methods which are useful to mainstream developers; and on the other, underpinning mainstream methods with formal foundations.

1 Introduction

Formal Methods are now becoming an established technology for software development in several sectors of the industry particularly for systems which are critical with respect to safety or finance. Particular domains where there has been a significant uptake are critical instrumentation and control systems in military and civil avionics systems, terrestrial transportation systems, nuclear control systems, and space systems. Although this is a large industrial market sector, estimated as 3 Billion Euro annually within Europe, there is still much work to be done if formal techniques are to become a major force in mainstream industrial software development.

The formal techniques which have been the subject of the research described in this paper have been available for some years, but technology users have been reluctant to fully adopt what has been offered on the market, as the supplied technologies lacked sufficient track record of use, were perceived as difficult and costly to use, and required highly specialised mathematical expertise in the development team.

In this paper, I give a synopsis of the results of a number of research projects which have made some progress towards providing evidence of the benefit of following the formal approach. This evidence concerns the quality of the delivered system and the cost of its development and is both qualitative and quantitative in nature.

The formal methods considered cover a variety of techniques which can be summarised as follows:

- the use of *Mathematical Models* for specification of software,
- the use of *Proof* for calculation of properties of the specified software,

- the use of *Stepwise Refinement* to ensure the preservation of these propeties by the design decisions made during development, and
- the generation of *Test Cases* from specifications to complement formal proof in verification of the implementation.

The formal methods employed in the projects covered by this paper are primarily the "model-oriented" formal methods including:

- VDM [33], the Vienna Development Method originally developed at IBM research laboratories in Vienna during the 1970s, and then consequently at Manchester University, the Technical University of Denmark, RAL and IFAD.
- Z [42], developed at Oxford University during the 1980s, and
- B [1], also emanating from Oxford and then developed in several industrial organisations including BP and GEC Alstrom during the 1980s and 1990s.

This paper is outlines three lines of research undertaken in the author's group at RAL, presents some results from one of those lines, and suggests areas for ongoing further work. The next section identifies the three key lines of research and outlines a number of the research projects undertaken over the last decade. Then Section 3 presents two of these projects in more detail and key results from some from them. The last section describes some ongoing work and draws some conclusions.

2 Three converging lines of research

This section outlines a number of research projects and identifies three areas of research which are key to enabling the uptake of formal methods. The three lines of research are:

1. the *Advancement of Formal Methods* through the development of tools and techniques supporting particular methods,
2. the *Technology Transfer of Formal Methods* through their application in developments in collaboration with industry, and
3. the *Formalisation of Industrial Methods*, in particular diagrammatic methods for Object Oriented Analysis and Design.

These three lines of research contribute respectively to the practical feasibility of using formal methods in real-life software development; to the body of evidence for the costs and benefits of adopting them; and to the reduction of the level of specialisation required by the personel employing them.

The projects covered are summarised in the following sections.

2.1 Advancement of Formal Methods

The first line of research is concerned with the advancement of formal methods themselves and the devlopment of technology to support their use. These projects developed tools and standards for formal methods, both prerequisites to the industrial uptake of a technology.

The Mural project (1986-1990) developed a VDM specification support tool and proof assistant. Working with Cliff Jones' group at Manchester University, this project produced a tool supporting VDM developments and a generic proof assistant supporting the construction of proofs arising in those developments [23][25][24]. The resulting tool was among the first to a use graphical user interface to a theorem proving assistant and has since inspired a number of projects involving user interface design for theorem provers.

Proof in VDM. The Mural project led to the definition of an axiomatic semantics for VDM and a "Practitioner's Guide" to proof using that semantics [15]. A number of case studies in proof using that axiomatic semantics were also developed [7]. It also led to a line of reserach into the use of read and write frames in operation decomposition[9][10][8].

The ZIP project was a collaboration between British Aerospace, British Petroleum, IBM, Logica, Praxis, the University of Oxford and the Rutherford Appleton Laboratory. This project developed a semantics and proof theory for Z, which forms the basis of the current Draft of the British and ISO Standards for Z.

Verification Techniques for LOTOS (1990-1992) was a study of the theory and practise of the verification of specifications in the ISO standard LOTOS language. RAL's contribution to this project was in the development of the ERIL term rewriting formal reasoning system[37] and in undertaking a case study which used LOTOS to specify parts of the graphics standard GKS[29]. The other partners in the project were Royal Holloway and Bedford New College, Glasgow University, and British Telecommunications.

2.2 Technology transfer of Formal Methods

The second line of research concerns the technology transfer of formal techniques into industrial practice. This has been achieved through three industrial collaborative projects which have employed and assessed the VDM and B methods. These projects provide evidence of the cost/benefit of the use of formal methods necessary to reduce the risk of their uptake.

The B User Trials project (1992-1995) was the first UK industrial project involving the B-Toolkit. It was a collaborative project between RAL, Lloyds Register of Shipping, Program Validation Limited and the Royal Military College of Science and played a major part in bringing the B-Toolkit up to industrial quality. RAL undertook two case studies [22, 41] comparing B with VDM and Z respectively, and investigated the methodology employed in the construction of proofs [19, 26]. A summary of the project appears in [11].

The MaFMeth project (1994-1995) was the first project to bring VDM and B together to exploit their different strengths. A collaboration with Bull Information Systems, it was "application experiment" assessing a methodology covering the whole life cycle combining the use of VDM for early lifecycle development with B for refinement and code generation. It demonstrated the commercial viability of the use of formal methods by collecting quantitative evidence of the benefits in terms of fewer faults made in development and early detection of faults. Qualitative experience is reported in [14] and a quantitative analysis of the results in [13]. In particular it was found that animation, test case generation and proof are all cost-effective ways to find faults in formal texts [12].

The Spectrum project (1997) was a feasibility study into the commercial viability of intergating the VDM-Toolbox and B-Toolkit. The evaluation was undertaken from three perspectives: the industrial benefit of using the combined tool, the technical feasibility of the combination of the two tools and the commercial case for the development of a combined tool. RAL's partners were GEC Marconi Avionics, Dassault Electronique, Space Software Italia, the Commissariat à l'Energie Atomique, the Institute of Applied Computer Science (IFAD) and B-Core UK Ltd.

Further details of the MaFMeth and SPECTRUM projects are given in the Section 3.

2.3 Formalisation of Industrial Methods

These projects concerned providing some formal underpinning to techniques already established in industry, thus providing a low-cost entry route to the uptake of formal methods.

The TORUS project sought to raise the level of reuse in industrial process control projects, by defining and deploying reuse based work processes together with supporting tools, thus improving the efficiency over the whole design lifecycle. the project employed the ISO standard data modelling language STEP-EXPRESS. As part of the work, we undertook investigations into the formalisation of this langauge, and made suggestions for clarifications to its semantics[20] [18]. The project was in collaboration with Cegelec Projects Ltd., FLS Automation, and Alcatel ISR.

Formal Underpinning for Object technology (FUOT) This research with Imperial College London and the University of Brighton undertook the formalisation of the conceptual basis of diagrammatic Object Oriented Analysis and Design notations such as UML using the "Object Calculus" as a semantic framework[17]. It also analysed the notations and typical development steps in order to suggest improvements to make these techniques more sound scientifically. The work also raised some issues about subsystems which are the basis of ongoing research [16] (see Section 4.2).

3 Results of Technology Transfer Projects

This section presents in more detail one strand of research undertaken in the second of these research lines: the technology transfer of formal methods. It describes two projects, MaFMeth and Spectrum which have both been concerned with the assessment of the use of a combination of the VDM and B Methods. The next section presents some key results from those projects.

3.1 Heterogeneous development using VDM and B

VDM and B are two of the most industrially used formal methods. Both are model-oriented methods for the development of sequential systems based on first order calculi and set theory. Both have a set of proof rules defined for formal verification and validation. Both have a formal semantics: for B this is defined in terms of weakest preconditions, for VDM it is denotational. Both have been used for a variety of applications and are supported by commercial toolkits.

VDM - The Vienna Development Method VDM's origins lie in the definition of programming language semantics in the 1970s, but for many years it has been used in systems development generally [33], and there is now an ISO standard [32] of the specification language VDM-SL. It has a rich set of data type constructors, augmented by invariant predicates. Functions and state-transforming operations can be defined explicitly using a large expression and statement language or implicitly in terms of precondition and postcondition predicates.

The VDM-SL standard includes a denotational semantics [36]. The semantics is based on the three-valued Logic of Partial Functions[34] which explicitly deals with definedness of expressions and requires the demonstration of well-typing for the substitution of equals. In particular, the strong type system supports static detection of many well-formedness errors. A published proof theory, described in [33] and in greater detail in [15], supports the validation of VDM-SL specifications through the discharge of proof obligations.

One area of weakness in VDM relative to B is its lack of generally agreed large-scale structuring. The standard contains a "informative annex" describing several alternative approaches to modules, including one implemented within the IFAD Toolbox, and other structuring proposals exist[31].

The IFAD VDM-SL Toolbox [30] is an industrial strength commercially available tool which supports the ISO VDM-SL notation. The Toolbox includes a syntax checker, static semantic checker, and a pretty printer generating LaTeX output. In addition, it contains a debugger, an interpreter, and a C++ code generator. It is also possible to perform test coverage analysis of specifications. An earlier tool for VDM, VDM Through Pictures [27] developed by Bull Information Systems, which supported the generation of VDM "skeletons" from Entity-Relationship style diagrams was used in the MaFMeth project.

Areas to which VDM has recently been applied include railway interlocking systems, ammunition control systems, semantics of data flow diagrams, message authentication algorithms, relational database systems and medical information systems. A directory of VDM usage examples is available[45].

The B Method The B-Method [1] represents one of the most comprehensive formal methods currently being promoted as appropriate for commercial use. A development of VDM and Z, Jean-Raymond Abrial originated **B** whilst at the Programming Research Group at Oxford University in the early 1980s, and subsequently at British Petroleum Research (BP) and DIGILOG. Supported forms are now commercially available from B Core Ltd. and Steria Technologies de l'Information.

The B-method employs the Abstract Machine Notation (B-AMN) which uses a notion of *generalised substitution* to represent state transformations, a style of specifying operations which is more "natural" to the programmer than the pre/post predicates of VDM. The B-method also has powerful structuring mechanisms based on a notion of *Abstract Machine* which offers data encapsulation allowing modular design and development of systems.

B's underlying semantics is grounded in weakest preconditions over untyped set theory and classical logic; the type system is correspondingly weak, and the distinction between type-checking and proof is blurred.

The B-Toolkit [6], developed by BP and subsequently by B-Core UK Ltd focuses on rigorous/formal design by supporting refinement from abstract specification through to imperative code. Tools exist for supporting static analysis (type-checking), dynamic analysis (animation), design documentation, proof of refinement and code generation. Another support system for B is Atelier-B [4], provided by Steria Mediterranée which allows similar functionality to the B-Core Toolkit.

Examples of the use of B include the development of communication security protocols, subway speed control mechanisms, railways signalling, executable database programs and IBM's CICS product. A directory of B is maintained at [5].

Co-use of VDM and B Although VDM and B have the same expressive power in theory, a comparison undertaken during the B User Trials project observed [22] that VDM encourages a style of specification where implicit invariants and explicit frames are employed with postconditions to describe operations as abstractly as possible, whereas the representation of operations with explicit invariants and implicit frames employed in B encourages overspecification and the introduction of implementation bias reducing possible non-determinism. This difference arises from the different focus of the two methods and has led to the development of different functionality in the supported forms of the methods. Figure 1 and Table 1 show the different phases of the lifecycle favoured by the two methods and the complementary features currently provided by each toolkit.

3.2 MaFMeth : The Measurement and Analysis of a Formal Method

The combination of VDM and B was first explored by RAL and Bull Information Systems in the MaFMeth Project[13, 12]. This project developed part of a transaction management system using VDM for the initial design and analysis, and then translating into B for development and code generation.

The formal development was part of the second release of an application integration product of the type often known as "middleware" which allows applications to communicate in a number of ways via a single application programming

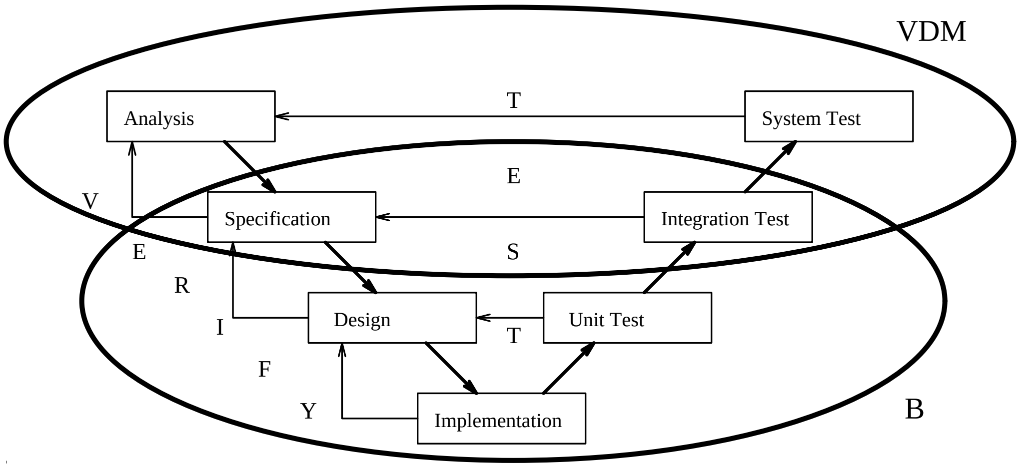

Fig. 1. The lifecycle identified for heterogeneous development using VDM & B

Task	IFAD VDM-SL Toolbox	B-Toolkit
Requirements capture	×	×
Visualisation	×	×
Abstract Specification	√	×
Type checking	√	∼
Prototype code generation	√	∼
Test coverage	√	×
Animation/Execution	√	√
Modularity	∼	√
Refinement	×	√
Proof	×	√
Final Code generation	×	√
Design documentation	×	√
Version Cntl/Config Mgmt	×	∼

Key
√ = good support
∼ = some support
× = no support

Table 1. The complementary functionality of the VDM and B Toolkits.

interface. Its primary function is to provide distributed, multi-platform, inter-application message handling services involving message routing, storage, transformation and enrichment transparently to the applications. The component of the system which was developed formally monitors the status and contents of the message queues and allows individual messages to be updated when required.

The project was undertaken in a conventional system software department with a development process certified as ISO9001 (TickIt) [44] compliant for its quality management system and operating at a point close to level 3 of the SEI Capability Maturity Model [40]. The development process adopted in this project was influenced by the desire to assess a variety of formal techniques covering as much of the development life cycle as possible, and the requirement that the resulting code had to be closely integrated with code developed by other methods.

The decision to employ a combination of VDM and B was motivated by the complementary facilities offered by the two toolkits and previous experience [22] which had shown the complementary strengths of the VDM and B methods. Three formal specifications were produced. The first, most abstract, specification was developed in VDM using VDM through Pictures. This was translated by hand into B Abstract Machine Notation, in order to conduct the first and second specification decomposition with the B-Toolkit, the result of which was then used to automatically generate C code.

Three forms of analysis were undertaken for validation and verification. *Animation* was used to validate the design during development, whereas post facto verification was undertaken using *test cases* and *proof obligations* which were generated from the specifications.

Measurements relating to these activities were taken in order to compare the formal development process with the conventional one used in that department, and to compare the relative effectiveness of the various stages of the formal process. For the former, the results of a number of development projects, all producing sub-products with similar characteristics, were compared using the department's existing programme of metrics. For the latter, faults were classified according to the development stage at which they were discovered and the stage at which they were introduced.

3.3 SPECTRUM : a step toward a unified method

Following MaFMeth, the SPECTRUM project also assessed the benefit of the combined use of VDM and B within a single development process. The project was a collaboration with four user partners in considering application domains:

- GEC Marconi Avionics considering avionics systems,
- Dassault Electronique considering terrestrial transport embedded control,
- Space Software Italia considering satellite communication control, and
- Commissariat à l'Energie Atomique considering nuclear plant control;

and two technology suppliers:

- the Institute of Applied Computer Science (IFAD) suppliers of the VDM-Tools, and
- B-Core UK Ltd. suppliers of the B-Toolkit.

The Rutherford Appleton Laboratory provided two roles:

- Expertese in VDM and B and their combination, and
- Neutral Project Coordination.

The SPECTRUM project had three objectives:

- to assess the cost-effectiveness for the user partners, of a development process employing an integration of the VDM and B formal technologies,
- to determine the technical feasibility of this integration; and
- to investigate the commercial potential for the tool suppliers of the integration of supported forms of VDM and B.

The first objective was addressed through the development of user scenarios exploring the utility of the various functions available from the two toolkits and the advantages of employing them in terms of development cost and product quality. The scenarios were in the application domains of avionics systems and terrestrial transport embedded control. They were also reviewed from the perspectives of satellite communication control and nuclear power plant control. Further details of the case studies can be found in [28], [43], [3].

The second objective was address through investigations into the feasibility of automating support for the translations between VDM and B. An approach to translation requiring the analysis of usage of types in the flat VDM specification in order to synthesise a useful decomposition into modules in B was proposed. Further details of the translations can be found in [38], [21]. This work formed a key input to the VDM+B project described in Section 4.1.

To address the third objective, the project assessed the commercial case for the provision of required functionality. It explored the overall cost-benefits of introducing the proposed method and the commercial viability of the proposed tool integration. It prepared a detailed market analysis and pricing policy for the combined tools. The results of this work are considered to be commercial in confidence.

3.4 Evidence of Error Reduction

The MaFMeth project undertook measurement and analysis of the total number of faults introduced and found during development. Despite the use of two notations and the lack of integrated tool support, quantitative analysis of the faults found at unit test shows the approach to be very effective both in cost and quality. Figure 2 compares data from this project with three others undertaken by the user partner using structured design. The four projects were all developed in the same environment over a period of about 3 years and all used a similar development process apart from the technology involved. All projects

	Project 1	Project 2	Project 3	MaFMeth
Application	System software utilities	Transaction monitor modifications	System software application	System software middleware
Approach	Yourdon	Yourdon	VDM / Yourdon	VDM / AMN
Size (LOC)	3000	1100	1300	** 3500
Effort (days)	65	80	27	43
Effort / KLOC	21.5	72.5	20.5	12.5
Faults at unit test	27	17	7	3
Faults / KLOC	9	15.5	5.5	0.9

*** Normalised against amount of library code used. (Total was 8000).*

Fig. 2. Comparison of numbers of faults found

were undertaken by engineers from the same development group and all were fragments of much larger developments. Similar testing procedures, based on manual identification of tests, were followed in each case. All, bar project 2, were new developments, whereas project 2 was a complex modification to an already heavily maintained system software component (hence, perhaps, the low productivity and quality of that development). None of the effort figures include the learning and technology transfer time which is inevitable in applying new approaches.

The LOC figure (Lines of Code) is clearly central to the metrics and, for projects 1 to 3, refers to C language statements. For MaFMeth, in all 8000 lines of code were generated. However much of this arose from library components. The figure of 3500 lines of code is the developer's estimate of the amount of code that would have been produced to implement the same functionality without attempting any reuse. In fact, 1200 lines of implementation level B notation were produced to generate the final C code.

The figures show that the MaFMeth project produced, on average, more code per day than any of the previous projects. Of course, this result must be tempered by the different application areas and the possible inaccuracy in the estimate of the equivalent number of lines of code. However, the improvement of nearly 100% is noteworthy.

Even more significant are the results concerning the number of faults at unit test. The unit testing used aimed at 100% functional black box test coverage and 100% branch level white box coverage. This was achieved by identifying test cases using techniques including equivalence partitioning, boundary value analysis and a judicious amount of error guessing! The MaFMeth project produced less than 20% of the faults of the next best project.

No attempt was made to moderate the effectiveness of fault finding by the sev the faults found. Such an analysis should contribute to an estimate of the cost-effectiveness of each activity.

Unfortunately, no figures for faults found during validation testing and customer use are available.

3.5 Evidence of Cost Reduction

Further evidence of cost reduction and comarison with costs for other development approaches was provided by GEC Marconi in SPECTRUM. Figure 3 shows an abstracted graph plotting the expected cost of using different formal methods in three projects over recent years.

The two horizontal lines represent the expected cost of employing their informal development process for saftey critical and non-safety critical system components.

The graph shows the falling cost of formal methods over recent years and in particular emphasises that the cost is now competative for safety critical systems and approaching viability for non-critical developments.

The vertical bars associated with the data points represent the high margin for error in prodictions made from a few small projects.

3.6 Relative Benefit of Formal Activities

The MafMeth project also undertook an analysis of the relative cost/benefits of each develepment activity. For these puposes the development process was divided into 13 activities, with varying degrees of tool support. These are depicted in Figure 4. The distribution of effort by activity is shown in Figure 5. Some activities, for example the initial B specification and its animation, are grouped together as they were carried out simultaneously and no separate effort figures were kept.

As might have been expected, the bulk of the design effort was in the main development in B. A substantial component was also expended on the early specifications in VDM. Very little effort was required during the testing stage.

The faults found can be plotted against these efforts as a histogram with the width of columns representing the relative effort expended in each stage (Figure 6). However, when inspecting this it must be remembered that some stages involved development whereas others purely involved review. For stages B1-2, one cannot assess how much effort was expended in finding faults through animation and how much on development, but if one assumes that approximately one half of this effort was spent on each activity, then the dotted line applies.

Note how the most efficient fault finding occurs during test generation, animation and proof. Although this can perhaps be attributed to the fact that most faults were found before testing occurred, the test generation and proof stages allow a different perspective on the specification and highlight problems which might otherwise be invisible to the developer.

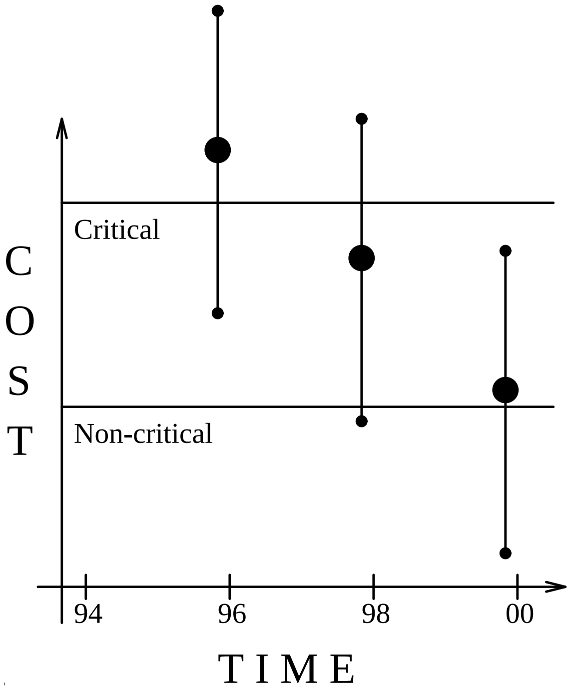

Fig. 3. Comparison of numbers of faults found

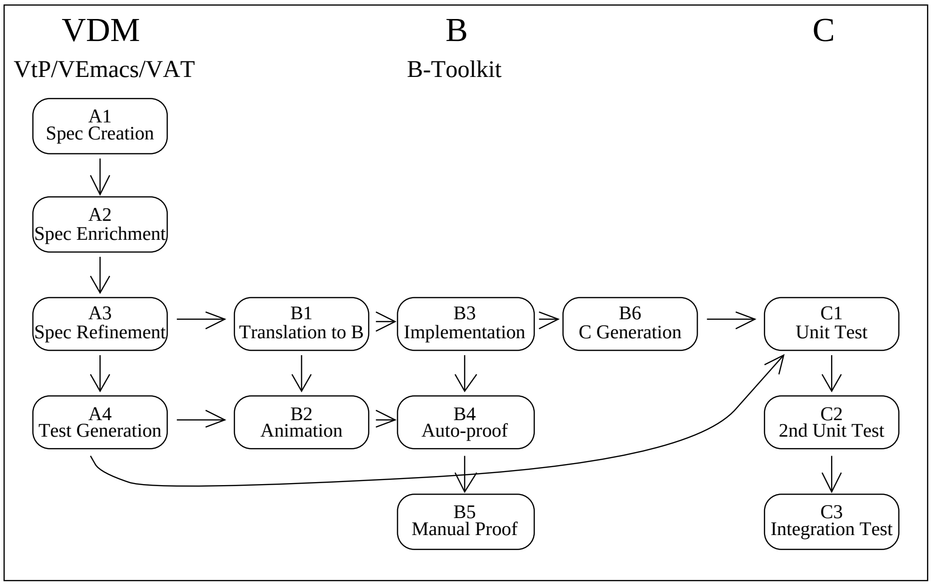

Fig. 4. Development activities identified in the MaFMeth project

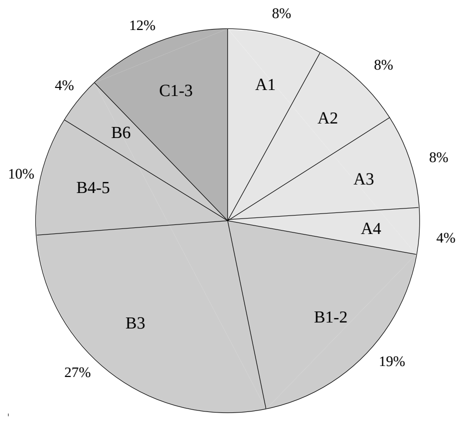

Fig. 5. Effort expended by each project stage

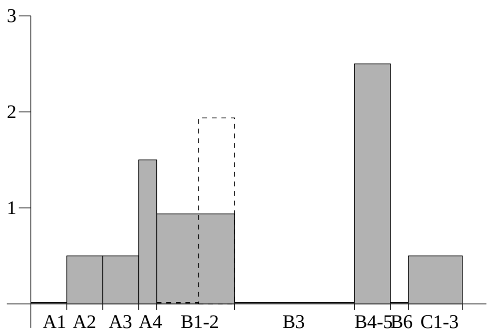

Fig. 6. Faults found per day by project stages

4 Ongoing work and Conclusions

This section outlines two current projects which are bringing together the three lines of research described above. The VDM+B project building on lines 1 and 2, the advancement and technology transfer of formal methods; and the Subsystems project building on lines 1 and 3, the advancement of formal and industrial methods, it then draws some overall conclusions.

4.1 The Integration of Two Industrially Relevant Formal Methods (VDM+B)

The VDM+B project (1998-2001) is the third project on VDM and B integration building on the results of the MaFMeth and SPECTRUM projects. In recognition of the pragmatic nature of the earlier work, the goal of this project is establish a formal foundation of heterogeneous development in VDM and B.

An obvious point of concern is the foundational differences in the languages. VDM is based on the 3-valued Logic of Partial Functions (LPF) whereas B is based on classical First Order Predicate Calculus. Work on developing proof support for VDM [2] has shown that in a framework with dependent types, such as PVS [39], most specifications which employ partial functions for their expressivity can be directly translated to functions which are total over a subdomain. The remaining uses of partiality represent a particular form of lazy concurrent disjunction which is built into LPF but not available in B.

Although the two notations are founded on a different logic, the proof rules in the B-Toolkit do have a flavour more akin to those of VDM where typing hypotheses are used as guards to the expression construct introduction and elimination rules. In the absence of a standard form for proofs that would enable proofs developed in one system to be checked with another, it is important for the certification of formal developments to be able to "second source" the theorem proving capability. This will allow proof support to be developed in a number of systems and contribute to the certification of theorem proving capability for use in safety critical systems. Current support for the languages has not been certified in this way.

A further area of difference is higher level modular structuring. Several approaches to modularisastion exist for VDM. The VDM standard language, VDM-SL, has no structuring mechanism although a form of modularisation is given as an informative annex, the IFAD VDM Toolbox supports a simple form of modules and VDM++ [35] has an object-oriented notion of structuring based on classes. On the other hand, the ability to incrementally present a specification is central to B where implementations can be constructed in a structured way by composing implementations of separate components.

Thus transformations between structured specifications in the two formalisms should, in some sense, preserve the locality of information. For example, in moving from a single module of VDM where the structure is based around a hierarchical definition of record types, we would hope to achieve a B specification which used machines to mirror the structure of the records. The danger is that in "coding up" such a complex refinement into the translation we risk the soundness of the translation. One possible approach [38] is for the translation to result in two levels of B specification and a refinement between them. In this way the translation is kept simple, whilst the complexity of the refinement is localised within the one formalism and hence more amenable to verification.

4.2 Objects, Associations and Subsystems: a hierarchical approach to encapsulation

The Subsystems project (1999-2002) arose from observations made in the Formal Underpinning of Object Technology project. That project observed that although subtyping and inheritance provide a hierarchical means of classification of objects, the class-instance paradigm is essentially flat and does not directly support the nesting of objects within objects. This led us to propose a notion of subsystem which generalises the class-instance-based concept of object, yielding an approach to system-specification employing object-like encapsulation in a nested hierarchy of components[16].

The strength of these subsystems lies in generalising key features of the success of object orientation. Objects provide a simple yet powerful basis for modularity through encapsulation. Aggregation of attributes in objects, and objects in associations, provides a basis for data-encapsulation; object identifiers globally identify instances and give an implicit indirection which distinguishes attributes which are themselves objects from attributes which are pure values. Objects can also provide a basis for establishing non-interference in concurrent implementations. On this basis, it seems that the OO approach would benefit from an old idea: hierarchical structuring.

In [16], we observed that the compositional interpretation of object-oriented designs requires the identification of theories intermediate between those of the constituent classes and associations and that of the entire system; and, how many constructions are naturally interpreted in theories corresponding to identified parts of the overall system. This project will investigate subsystems as first-class objects in OO system description achieving a hierarchical form of object-orientation.

4.3 Conclusions

We have presented some key results of two projects measuring the costs and benefits of the formal approach. This evidence indicates that formal methods are currently cost-neutral for safety critical systems and deliver higher quality than alternatives. The evidence contributes to the case upon which the adoption of formal techniques should spread beyond those applications domains where formal techniques and mandated or highly recommended such as defense and transport (c.f. UK Def-Stan 00-55 and French RATP requirements) to other safety and financially critical applications.

We have indicated the falling cost of formal methods which are now becoming competative with other methods used in non-critical systems development.

Further cost reduction, in particular below the cost of systems development for non-critical systems, will increase the market enormously.

However, conclusions drawn from these projects should be moderated by the small size the developments and the fact that the development teams were also small and staffed by self-selected individuals who, being keen to make a success of the experiments, were perhaps better motivated than average. It would not be wise therefore to extrapolate these results to larger projects.

Despite these qualifications, there is some evidence in these results in favour of formal methods. Faults are inevitable and their detection is aided by formalisation. It seems that any analysis, whether animation, proof obligation generation, proof, or testing, is worthwhile. These activities are only possible once the objects involved are formalised.

The three lines of research described provide three-legs of support for industrial strength formal methods: improved methods and support for them; evidence of the costs and benefits of their use; and formal underpinning of established industrial methods.

This work has contributed to an accumulation of evidence for the benefits of formal methods. It has raised awareness of the need to gather such evidence for larger projects and has demonstrated some techniques for doing so.

References

1. J-R. Abrial, *The B-Book: Assigning Programs to Meaning*, Cambridge University Press, ISBN 0-521-49619, 1996.
2. S. Agerholm, Translating Specifications in VDM-SL into PVS, in Higher Order Logic Theorem and Its Applications: 9th International, LNCS 1125, Springer Verlag, 1996.
3. S. Agerholm, P-J. Lecoeur and E. Reichert., *Formal Specification and Validation at Work: A Case Study using VDM-SL* Proceedings of Second Workshop on Formal Methods in Software Practice, Florida, ACM, March 1998.
4. Atelier B, Steria Mediterranée, The B Page URL, <http://www.atelierb.societe.com/PAGE_B/uk/bhomepg.htm>
5. The B method (virtual library page) `http://www.comlab.ox.ac.uk/archive/formal-methods/b.html`.
6. B-Core (UK) Ltd, *B-Toolkit User's Manual, Version 3.0*, 1996. For details, contact Ib Sorensen, B Core (UK) Ltd, Magdalen Centre, Robert Robinson Avenue, The Oxford Science Park, Oxford OX4 4GA. Tel: +44 865 784520. E-mail: Ib.Sorensen@comlab.ox.ac.uk, WWW: http://www.b-core.com/
7. J. Bicarregui (ed.) *Proof in VDM: Case Studies*, Springer-Verlag, FACIT series, 1998.
8. *Intra-Modular Structuring in Model-Oriented Specification: Expressing Non-Interference with Read and Write Frames.*
   J.C. Bicarregui. Ph.D. Thesis, University of Manchester (UMCS-95-10-1).
9. *Operation Semantics with Read and Write Frames.* J.C. Bicarregui. Proceedings of the 6th Refinement Workshop, David Till (Ed.), Springer Verlag.
10. *Algorithm Refinement with Read and Write Frames.* J.C. Bicarregui. Proceeding of Formal Methods Europe '93, Woodcock and Larsen (Eds), LNCS 670, Sp ringer-Verlag.

11. *Formal Methods Into Practice: case studies in the application of the B Meth od.* J.C. Bicarregui *et al.* **I.E.E. proceedings software engineering, Vol. 144, No. 2**, 1997.
12. *Making the most of formal specification through Animation, Testing and Proo f.* J.C. Bicarregui, J. Dick, B. Matthews, E. Woods. **Science of Computer Programming, Vol. 29.** (1997), Elsevier Science.
13. *Quantitative Analysis of an Application of Formal Methods.* J.C. Bicarregui, J. Dick, E. Woods. Proceeding of FME'96, Third International Symposium of Formal Methods Europe, LNCS, Springer-Verlag.
14. *Supporting the length of formal development: from diagrams to VDM to B to C .* J.C. Bicarregui, J. Dick and E. Woods. "Proceedings, 7th International Conference on: Putting into practice method and tools for information system design", Nantes (France), October '95, IUT de Nantes, H. Habrias (Ed) ISBN 2-906082-19-8
15. *Proof in VDM – A Practitioner's Guide.* J.C. Bicarregui, J.S. Fitzgerald, P.A. Lindsay, R. Moore and B. Ritchie. ISBN 3-540-19813-X, FACIT, Springer-Verlag
16. *Objects, Associations and Subsystems: a hierarchical approach to encapsulat ion,* J.C. Bicarregui, K.C. Lano and T.S.E. Maibaum, Proceedings of ECOOP'97, 11th European Conference on Object-Oriented Programming , Jyvaskyla, Finland, June 1997.
17. *Towards a Compositional Interpretation of Object Diagrams.* J.C. Bicarregui, K.C. Lano and T.S.E. Maibaum, in "Algorithmic Languages and Calculi", Proceedings of IFIP TC2 working conference, Strassbourg, Feb. 1997, Bird and Meertens (Eds), Chapman and Hall. 0-412-82050-1, September 1997.
18. J.C. Bicarregui, B.M. Matthews, *The specification and proof of an EXPRESS to SQL "compiler"*, in Proof in VDM: Case Studies, J.C. Bicarregui (Ed), FACIT series, Springer-Verlag, 1998.
19. *Formal Methods in Practice: a comparison of two support systems for proof.* J.C. Bicarregui, B. Matthews. SOFSEM'95: Theory and Practice of Informatics, Bartosek et al. (Eds), LNCS 1012, Springer-Verlag.
20. J.C. Bicarregui, B.M. Matthews *Formal perspectives on an Object-Based Modelling Language* Proceedings of the 6th EXPRESS User Group Conference, Toronto, Canada. IEEE Computer Society, ISBN 0-8186-8641-3, 1996.
21. *Investigating the integration of two formal methods*, Juan Bicarregui, Brian Matthews, Brian Ritchie, and Sten Agerholm, **Formal Aspects of Computing, Vol. 10.** pp. 532-549, Springer-Verlag, Decemb er 1998.
22. *Invariants, Frames and Postconditions: a comparison of the VDM and B notati ons.* J.C. Bicarregui and B. Ritchie. Proceeding of Formal Methods Europe '93, Woodcock and Larsen (Eds), LNCS 670, S pringer-Verlag. b. 1995.
23. *Reasoning about VDM Developments using the VDM Support Tool in Mural.* J.C. Bicarregui and B. Ritchie. Proceeding of VDM 91, Prehn and Toetenel (Eds), LNCS 552, Springer-Verlag.
24. *Supporting Formal Software Development* and *The Mural VDM Support Tool.* J.C. Bicarregui and B. Ritchie. in "Mural, A Formal Development Support System." Jones, C.B. *et al.* Springer-Verlag, ISBN 3-540-19651-X.
25. *Providing Support for the Formal Development of Software.* J.C. Bicarregui and B. Ritchie. Proceedings of the First International Conference on Systems Development Environments and Factories. Madhavji, Schafer and Weber (Eds), Pitman.
26. *Experiences with Proof in a Formal Development*, D. Clutterbuck, J.C. Bicarregui and B.M. Matthews, Proceeding of 1st International Conference on B, Institut de Recherche en Informatique de Nantes, France, November 1996.

27. A.J.J. Dick and J. Loubersac. *A Visual Approach to VDM: Entity-Structure Diagrams,* Technical Report DE/DRPA/91001, Bull, 68, Route de Versailles, 78430 Louveciennes (France), 1991.
28. J. Draper (ed) *Industrial benefits of the SPECTRUM approach* SPECTRUM Project External Deliverable 1.3, 1997. See http://www.itd.clrc.ac.uk/Activity/SPECTRUM.
29. D A Duce, F Paterno, *Lotos Description of GKS-R Functionality*, Formal Methods in Computer Graphics Workshop Eurographics Association, 1991.
30. R. Elmstrøm, P.G. Larsen, and P.B. Lassen, *The IFAD VDM-SL Toolbox: A Practical Approach to Formal Specifications*, ACM Sigplan Notices 29(9), pp. 77–80, 1994. For more information see http://www.ifad.dk/
31. J. S. Fitzgerald, *Modularity in Model-Oriented Specification and its Interaction with Formal Reasoning.* Ph.D. Thesis, University of Manchester, 1991.
32. ISO *Information Technology - Programming Languages - Vienna Development Method-Specification Language. Part 1: Base Language.* ISO 13817-1, 1996.
33. C.B. Jones, *Systematic Software Development using VDM.* Prentice Hall, second edition, 1990.
34. C.B. Jones and C.A. Middelburg, *A Typed Logic of Partial Functions Reconstructed Classically,* Acta Informatica, 1994.
35. K. Lano, S. Goldsack, Integrated Formal and Object-oriented Methods: The VDM++ Approach', 2nd Methods Integration Workshop, Leeds Metropolitan University, April 1996;
36. P. G. Larsen and W. Pawlowski, *The formal Semantics of ISO VDM-SL,* Computer Standards and Interfaces, Volume 17, numbers 5-6, 1995.
37. B.M. Matthews, *MERILL: An Equational Reasoning System in Standard ML*, 5th Int. Conf. on Rewriting Techniques and Applications p.414-445 C Kirchner (Eds), LNCS 690, Springer-Verlag, 1993.
38. *Synthesising structure from flat specifications*, B. Matthews, B. Ritchie, and J. Bicarregui, Proc. of the 2nd International B Conference, Montpellier, France, April 22-24, 1998. Springer Verlag, LNCS 1393.
39. S. Owre et al. PVS: Combining Specification, Proof Checking, and Model Checking, Computer-Aided Verification, CAV '96, Rajeev et al (Eds) Springer-Verlag LNCS 1102, 1996.
40. M.C. Paulk, W. Curtis, M.B. Chrissis, C.V. Weber, *Capability Maturity Model for Software*, Version 1.1, Carnegie Mellon University Software Engineering Institute Technical Report, CME/SEI-93-TR-24, February 1993.
41. *Experiences in Using the Abstract Machine Notation in a GKS Case Study.* B. Ritchie, J.C. Bicarregui and H. Haughton. Proceeding of Formal Methods Europe '94, Naftalin, Denvir and Bertran (Eds), LNCS 873, Springer-Verlag.
42. J.M. Spivey. *The Z Notation, A Reference Manual.* Prentice Hall, ISBN 0-13-983768-X, 1987.
43. H. Treharne, J.Draper and S.Schneider, *Test Case Preparation Using a Prototype*, Proc. of 2nd Int. B Conf. B'98: Recent Advances in the Development and use of the B Method, LNCS 1393, Springer Verlag, April 1998.
44. U.K. Department of Trade and Industry, *TickIT: Guide to Software Quality Management, System Construction and Certification using ISO9001/EN29001/BS5750 Part 1*, TickIT Project Office, 68 Newman Street, London, W1A 4SE, UK, February 1992.
45. The VDM examples repository: `http://www.ifad.dk/examples/examples.html`