FORMAL METHODS
EUROPE

FACS Europe

BCS
FACS

The Newsletter of the BCS Formal Aspects of Computing Science Special Interest Group and Formal Methods Europe.

Series I Vol. 1, No. 2 Spring 1994

Contents

Editorial

Welcome to the Spring issue of FACS Europe, hope you enjoyed reading the last bumper issue. This issue has a comprehensive range of contributions beginning with a communication from Cliff Jones which responds to Anthony Hall's comments on the magic roundabout tour of comparing Z and VDM. We also introduce a much needed Education Column which should enable those of us involved in the teaching of formal methods to exchange and share ideas on curricula and pedagogy; those of us who like to solve problems should find the 'Notes and Queries' an inviting challenge. Our regular RAISE Column invites participation on RAISE standardisation.

Two contributions that provide useful and easy to digest information are the Formal Methods Tools Database and the Formal Computer Science in JFIT. The first contribution informs on an initiative to set up a database, invites participation and provides a summary of existing tools. The second contribution reports on the number of applications submitted for the Formal Computer Science area under JFIT in 1992/93; their total value, the number funded etc, it also provides details of the funded projects.

Our roving reporters have submitted interesting reports on key areas including formal aspects of object orientation and term re-writing.

Former FACS FACTS readers might be wondering about the whereabouts of F-X REID. The editors have not received any correspondence from him/her although the column on the formal specification of neurosis in the last edition seemed to carry that hallmark. We have had a lot of interest in people wanting to obtain copies of FACS Europe from all over the world and would tell anyone not on the mailing list to join FACS by contacting the FACS membership secretary in order to receive a copy. Our mailing list is well over 700!

Finally, the success and quality of FACS Europe is in your hands, the more contributions we receive the higher the quality of the newsletter. We welcome all types of contributions on formal aspects: technical, informative, humorous; reports and much more. We look forward to hearing from you.

Jawed Siddiqi

Acknowledgements

This edition was compiled, edited and LaTeX'd at the Computing Research Centre of Sheffield Hallam University, by:

Brian Monahan	University of Manchester
Chris Roast	Sheffield Hallam University
Jawed Siddiqi	Sheffield Hallam University

VDM and Z again: A Reply to Anthony Hall's Response

Cliff Jones
cbj@cs.man.ac.uk

March 24, 1994

This 'letter to the Editor' is written in reply to Anthony Hall's *A Response to Florence, Dougal and Zebedee* pp 31–32 of *FACS Europe*, Vol. 1, No. 1. I should first like to welcome Anthony's basic point: there is a need for another article. In defence of what *was* written in the original paper (*Understanding the Differences between VDM and Z* pp 7–30, *FACS Europe* Vol. 1, No. 1.) it could be pointed out that its authors have often been asked about differences between Z and VDM[1] by specialists who have found the topics which are addressed in our paper to be very much the ones which interest them. The alternative proposals (in the logics for Z and VDM) for dealing with partial functions, for example, is a topic of continuing research interest. It is certainly clear that users of specification languages may well have other, or at least additional, concerns and another paper could be written which would be more suitable for this audience. We would certainly encourage someone to write such a paper and therefore welcome the main thrust of Anthony's letter.

I should like to offer some comments which might influence someone undertaking the envisioned article: I address the issue of objectivity and provide some technical reminders. For brevity below I will refer to the original paper as MRP (Magic Roundabout Paper) and to Anthony Hall's letter as AHL.

The authors of MRP come (two) from a Z background and (one) from a VDM background and it was intended to be a strength of the paper that they managed to agree a combined text. It should have limited any extreme claims from one side to which the other authors felt they would not wish to put their name. Let me then now make clear that I am changing to the first person singular. I will again try to retain a spirit of fairness (and I hope my co-authors would not disassociate themselves from much of what follows). In some places, I certainly put a VDM point of view (where I feel that Anthony has listed arguments in favour of Z) but I hope that this will be seen as a contribution to understanding the differences rather than fuel for a battle between two valid approaches.

I must first address the point in AHL about the lack of a discussion of modelling style in MRP. The reason that this is not covered is that one can make different modelling decisions when writing in a specification in Z or VDM and these differences have very little to do with the differences between the Z and VDM languages. For example, Z specifications tend to introduce 'derived' state components whose values are constrained by invariants (they are redundant in the sense that their values are completely determined by other state variables); in VDM specifications one would normally find these values being computed by auxiliary functions (thus keeping to a minimal state). There is no technical obstacle to reversing this stylistic trend and it would therefore be confusing to labour the point in a comparison between Z and VDM specification notations. Of course the differences between models can be crucially important to their users and I personally find this a fascinating topic (I have just taught an MSc course exactly on Abstract Models of Computer Systems; my examples have been taken from both the Jones/Shaw and Hayes case study books); but this is not the subject of MRP.

[1]Indeed, the strongest reason for writing this paper was precisely that this question has come up many times in the standardisation activities; members of some foreign ISO committees would ideally have liked to have seen just one specification language coming forward for standardisation.

AHL does challenge the technical point as to whether VDM can 'just be extended'. Personally I fought against seeing VDM as a fixed language carved on tablets of stone; I wanted very much more to see it as an evolving school of specification ideas. I have experimented with a variety of extensions (Anthony is kind enough to draw attention to my rely-guarantee approach to concurrency) and certainly do not believe that VDM-SL is the last word in specification languages. In fact, the whole issue of whether we should try to standardise specification languages is one about which I could write another long letter.

Focusing on AHL's specific challenge about the use of a new construct like general relations, I should have little hesitation in defining this as a new type if I wanted it for a large application and providing its foundations in a similar style to the foundations of other parts of VDM. In fact, in many cases it is possible to provide suitable extensions by writing a series of auxiliary functions and then one is 'just' faced with a concrete syntax problem as to whether those functions are written in normal functional notation or as infix operators.

I find the related claim in AHL that 'functions are relations (are sets)' is a good idea somewhat debatable. *This* is certainly an issue on which *I* should like to hear user's views. In my experience, very little 'comes for free' and I fear that there will be proof obligations when, say, composing functions with lists and claiming that the result is a list which are non-trivial and perhaps not understood by all users. But, I should concede that I do see the advantage of having general relations available as a type in a specification language and it is certainly an example that I should reconsider were I to be writing another book on VDM. The division between types is more rigid in VDM (than in Z) and this is connected with its original aim to facilitate the description of the denotational semantics of programming languages. (The extension mechanisms of Z could not easily be used to introduce Domain Theory.) In order to compare like with like, MRP does not address this application of the notations. But I have to say that, even if I were designing a non-domain theoretic specification language, I should probably retain the distinctions between types based on my best guess of what users can be expected to write safely.

There is no doubt in my mind that modularisation is still an open issue in the sense that I have seen no fully satisfactory solution (see the paper in MRP cited as [FJ90] and the responses in *Formal Aspects of Computing* Vol. 4 No. 1; John Fitzgerald and I still need to write a response to those 'solutions'!). I certainly include in my claim that there is no fully satisfactory solution among those modularisation proposals which have been written for VDM-SL and I echo Anthony's point that none of them have acquired the same degree of practical use as the 'flat' VDM specification language. But I have to add that I am pleased to hear that VDM-SL modularisation proposal did not work for 'shared states': it wasn't intended to allow such usage. Of course one can model something like shared state with a fairly gruesome construction of pointers. But, having said that this solution is not pleasing, I have to repeat my point that I know of no elegant solution to this particular difficulty.

I find very interesting the claim in AHL that 'the strengths and weaknesses of the Z schema are pretty well understood'. I am certainly prepared to believe that, in careful hands, the Z schema notation and in particular schema operators can be used to provide delightfully readable specifications. (For the first several years of the CICS specification work, I was a consultant on the project and reviewed many of the Z specifications; I developed great enthusiasm for the way Ian Hayes in particular was using the schema notation to present such specifications. I was, however, always left with the feeling that to be really sure I knew what the system was intended to do, I needed to have a 'flatten button'.) But I am not aware of any document which sets out warnings about the schema notation. (I haven't had the pleasure of attending one of Anthony's courses on Z.) I believe there are dangers in the way Z schemas can share names. This is borne out by reading a number of other Z specifications including those which are published in the literature as presumably examples of good style. I won't pursue this point here, both because it would only be fair to do so if I provided detailed examples and because I am anxious not to have a competition between the notations. One of the main purposes of MRP was precisely the reverse: we wanted to foster co-operation between different formal methods communities by explaining the essential differences rather than having people argue about irrelevant issues. But in connection with this issue I must point out that 'Z needs a modularisation mechanism' appears in the *co-authored* MRP.

The use of the schema calculus is certainly another area where I should like to understand user experience. How are schema combinators really used in practice? Are they just used to separate out exception conditions or do people really make many connections between free names in different schemas? AHL is emphatic that this is a good way to write specifications. With care I am sure this is true but it is unclear to me that you cannot get the same advantage by tasteful use of sub-functions without the dangers inherent in linking free names.

There is one other point on specifications which I should like to pick up for fear that a statement in AHL be taken as a complete rebuttal of claims in MRP. I believe that the separation of a pre-condition in a specification is a *specification issue.* In my experience of reading many large (informal) industrial specifications, people are actually better at describing the intended function (post-condition) than they are at recording the assumptions they are making about their system (pre-condition). It is therefore not just a technical point that suits the VDM development method to have a separate pre-condition; it is my firm opinion that forcing people to focus on a separate pre-condition when they are thinking about the specification is essential. I do not believe that writing down a post-condition and then 'computing the pre-condition' is a substitute for asking the user to document their assumptions.[2]

A major issue which is not addressed in MRP is development methods. Clearly, this would not have been easy to do since there is not – to my knowledge – a generally agreed development method associated with Z which was originally seen solely as a specification language. Of course, the fact that operation decomposition and data reification *were* addressed early in the evolution of VDM, has influenced the language. (I find it interesting how many of the decision's in Jean-Raymond Abrial's 'Abstract Machine Notation' are closer to VDM than to Z. I am certain that this is because AMN is seen as part of a development method. One could also observe that Carroll Morgan's 'refinement calculus' separates pre-conditions.) But the decision to exclude development methods from MRP did make it harder to motivate some of the differences.

Tony Hoare (see footnote 3, page 15 of MRP) appears to argue that one can undertake a specification in Z and then use VDM as the development method. This could be a masterful compromise but it comes up against exactly the sort of technical difficulties which are discussed in MRP. I certainly do not dispute that there could be a Z-like specification language which was more convenient for analysing requirements and generating the first specification and that a somewhat different (VDM-like?) notation or presentation of specification might be useful for recording the specification which is to be used from the beginning of the (VDM-like) design process. But is is clearly imperative that there are no gratuitous technical differences between these two languages. MRP has discussed issues like the appropriate logic to handle partial functions; if we are to come up with a 'beautiful pair of twins' the issues which are raised in MRP need to be resolved.

I will end as AHL ends: in summary I certainly agree that another article is waiting to be written. I personally enjoyed the collaboration in writing MRP and I believe the choice of authors from the different approaches resulted in a fairly balanced paper. I hope that, when it is written, the article written from the users' point of view is also fairly balanced.

Cliff Jones
1994-02-04

[2]During January there have been two relevant debates on `comp.specification.z`: One concerned schema composition which would benefit from identifying a pre-condition; in the other a Z user explained why a style which distinguishes (in some way) a pre-condition is to be recommended for avoiding short but cryptic specifications.

The Education Column

Roger Carsley
University of Westminster
155 New Cavendish Street
London, W1M 8JS
roger@westminster.ac.uk

Welcome to the Education Column, a column, it is hoped, to interest all — practitioners as well as academics.

The arena of education is particularly important to Formal Methods. Staff in education and industry work closely together, even exchanging places. The universities are educating the future recruits and offering advanced and training courses. At this stage in its maturity, all are undergoing a process of continuing education. In addition to the high profile and big money joint research projects, we also have a mutual interest in the more modest levels of the transfer (exchange!) of skills and development of the subject.

What is happening in education? What should be? What knowledge and skills are required by practitioners? What should be being taught now to prepare for the future? How can we best teach our subject?

This column can be a forum for wide-ranging discussions about Formal Methods Education. But a column is only its articles and correspondence and these are provided by participation. Please accept this invitation to share your views, experience and knowledge.

One special request for material for this column goes to those outside academia and the research organisations: How well do skills currently taught match needs? What is most relevant? What appears least relevant? Which areas would benefit from continuing education courses? What additional skills are desirable in future graduates?

Additionally, the following eclectic list (a pedagogic device) of potential topics and titles was extracted from a confused and incoherent 'customer'. Use any appropriate methods, preferably formal, (possibly induction, interpolation or intoxication) to infer a model of the requirements of the client. Hence, write and submit a report satisfying any aspect of the client's needs of your choice:

- reviews of books, software tools and other materials from the point of view of their value as teaching aids
- examples of 'successful' student projects
- "How not to choose a Formal Methods PhD topic/supervisor"
- challenges to a prevalent view (c.f. the NPL report "Formal Methods: A Survey" 1993) that the staple food of formal methods education is VDM/Z, spiced with a dash of concurrency
- experiences with 'alternative' forms of assessment
- "Do we need and do we have time for Modal Logic?"
- those elusive stimulating case studies and insightful examples
- curricula for the new millennium

- on the basis that we will have succeeded in our mission when FM is applied in other topics, examples of such
- "Hacking in Specification Language X: a student's guide"
- "A Comparison of Approaches to FME in Europe, the USA and Japan"

[this column has *no* travel budget! Ed.]

We launch the column, with grateful thanks to Dan Simpson, surveying research under JFIT thereby illustrating the scale, range and prominence of formal methods activity within the U.K. JFIT, the Joint Framework for Information Technology, a collaborative research effort involving industry, government and academia in the U.K. is supported by the Department of Trade and Industry (DTI) and the Science and Engineering Research Council (SERC).

May we also draw your attention to, if you do not already know of it, "Educational Matters" coordinated by Hans-Jörg Kreowski in the EATCS Bulletin. Currently, the focus there is on the mathematical education of software engineers and will follow responses to papers by David Parnas and Jacques Printz.

The names of those submitting the best work will be displayed in **boldfont** along side their efforts in the next column. Late submission is not permissible. Contributions should be sent to to the author, preferably by e-mail.

Notes and Queries Column

Anne Wright
annw@uk.ac.uclan.sc

We have recieved two related "queries" from Tim Denvir to which answers are sought.

Q1. What is the relationship between the formal semantics of a language and its proof theory?

To elaborate this question: the proof theory of a language is a deductive scheme for propositions about sentences in the language. For example the proof rules of an imperative programming language embody a language of propositions of the form $P\{S\}Q$ where P and Q are predicates and S is a statement in the language. This is probably the same question as:

Q2. Given a calculus, how does one find the algebra?

There seems to be a common evolution in the development of semantics for languages. The language is first defined by means of a calculus, for example the CCS rules over action-labelled relations and the Hoare-Floyd rules of imperative languages, then several years later an equational characterisation is produced which, one hopes, captures the same class of models. Can one find a general heuristic for translation between the two?

Please send replies and ripostes to the columnist.

RAISE Column

Concrete is more abstract

Chris George, CRI
cwg@csd.cri.dk

The implementation relation in RSL is intended to allow you to replace a module with an implementation of it in a larger context. This is what allows you to do 'development in the large'. If module B0 uses A0, and I have an implementation A1 of A0, then creating B1 from B0 by making it use A1 instead of A0 will give implementation of B0 by B1.

This requirement on implementation means that we have to be careful about using concrete types. For instance, if in A0 in our example we had a type definition

type T = Elem-**set**

then we cannot just write in A1

type T = Elem*

since the attempt to create B1 will generate type errors.

This may seem like a problem that can be circumvented, but it is part of a deeper problem. Implementation in RSL is based on theory extension: all the properties of a module must hold in any implementation. This is important for rigorous development. We want to show that our initial specification has the properties to meet the requirements. From then on we should be able to extend, to add properties, but not to change them. The theory of lists is not an extension of the theory of sets.

To avoid the problem of concrete types RSL allows sort definitions and axiomatic specifications. These are not always so easy to write, but there is general methodological advice. You identify the 'type of interest' and decide on signatures for your functions. Then they are easily categorised as generators or observers according to whether their result types depend on the type of interest or not. Then usually some observers can be defined in terms of others, they become 'derived'. Then for the non-derived observers you write an axiom for each observer-generator pair and each observer-constant pair.

This approach gives you the left hand-sides of all the axioms, with relative completeness (you can observe any finitely generated term), without overspecification, and, with reasonable care, consistency. It all sounds easy enough. But in practice it isn't. It turns out that some right hand sides are hard to formulate.

Consider for example the generic abstract data type example, the stack. 'is_empty' is probably an observer and 'pop' a generator, whether it just reduces the stack or also returns the top element. We will assume the former for simplicity, so we will also have an observer 'pop'. Now according to our method we will need axioms of the form

[is_empty_pop] is_empty(pop(s)) $\equiv$ ...
[top_pop] top(pop(s)) $\equiv$ ...

Any suggestions for the right hand sides? You might decide to add another observer, like 'depth', say, to deal with the first. The second seems to need yet another observer. It seems that the 'observer-generator axioms only' style may be difficult.

The standard approach is to say that, as with observers, some generators can be derived from others. If we take 'empty' and 'push' as the basic generators then we can define the generator 'pop' for non-empty stacks by [pop_push] pop(push(e,s)) ≡ s

(How to write [pop_empty] is another issue that is not relevant to this discussion). We can also go further (in RSL or in Larch, for example) and state that 'empty' and 'push' will be the only generators (i.e. we won't add any later in development) so that we can do induction.

The standard objection from the model-based school is now "This may work for stacks but what about queues. You never show those, do you." Well, we can show the corresponding axiom for a queue:

[deq_enq]
  deq(enq(e,q)) ≡
    if is_empty(q) **then** empty **else let** q′ = deq(q) **in** enq(e,q′) **end end**

but I am not convinced we can argue that it is easy or obvious.

But there is a more fundamental objection to [deq_enq] than its difficulty, and this also applies to [pop_push]. The problem is that such axioms preclude some obvious implementations. The standard example for a (bounded) stack is an array with a pointer. Pushing an element increments the pointer and inserts the element; popping the stack should involve just decrementing the pointer. But this won't give you the equivalence in the axiom [pop_push] unless you also clear the value above the pointer to some standard value. This is a waste of time, as this value can never be accessed again by the stack operations. Similar problems arise with the natural implementation of a (bounded) queue as a circular buffer.

Our abstract specifications are not sufficiently abstract: they preclude some implementations that, I hope, we would want to regard as correct. What is to be done?

This is not a new problem. Some people have attempted to solve it by an abstraction mechanism, a mechanism that weakens the equivalence in axioms like [pop_push] to an observational equivalence: 'stacks are equivalent when there are no observers that can distinguish them'. The problem with this is that in general there seems to be no way to finitely present the resulting theory. As a consequence the implementation relation in RSL would not have a finite expansion. (This is why **hide** in RSL is purely syntactic rather than behaving like an abstractor.) It is also not clear how to combine it with development: the meaning of the observational equivalence is likely to change if we add new observers.

My solution is based, ironically, on using concrete types. It seems natural to model both stacks and queues (as it happens) as lists. But I don't, for reasons I outlined above, want to say

type Queue = Elem*

What I do instead is to say there should be an observer that can observe any queue as a list. It should be clear that this will be true of my circular buffer implementation. I believe it to be so for any implementation of a queue. My observer will be hidden because I don't want to expose anything about the model I am using to users of my module, and in fact I intend not to implement this observer in the final program: it is just a specification device. So I start with

type Queue
value list_of : Queue → Elem*

Now it should be clear that all the other observers, like 'is_empty', become derived:

value
is_empty : Queue → **Bool**
is_empty(q) ≡ list_of(q) = ⟨⟩

This means that when I follow the observer-generator paradigm I only need to define axiomatically the relations between my one observer 'list_of' and the generators:

[list_of_empty] list_of(empty) ≡ ⟨⟩,
[list_of_enq] list_of(enq(e,q)) ≡ list_of(q) ^ ⟨e⟩,
[list_of_deq] list_of(deq(q)) ≡ **tl** list_of(q) **pre** ~is_empty(q)

(I have presented the axioms for an unbounded queue for simplicity, but the bounded version is only very slightly more complicated.)

Now I have axioms that are easier to write (and get right) and expressions using the convenient operators for lists. The axioms only relate observers and generators, so I also have a specification based on a concrete type that is demonstrably more abstract (in the sense of permitting more implementations) than the traditional 'abstract' version.

RAISE Standardisation

Maurice Naftalin, Lloyd's Register
`tcsmpn@aie.lreg.co.uk`

RAISE has finally set forth on The Longest Journey - standardisation. We regard standardisation as important not only for the industrial credibility of RAISE, but also because it provides a guarantee to the outside world that the definition of the language and tools have been subjected to careful independent scrutiny. So our first action has been to convene a proto-Review Panel, which will have the responsibility of transforming the existing RSL documentation to standard-ready form - although we anticipate that there may be quite a lot of new work to do as well. We are also asking existing RAISE users for suggestions for changes to the language to be standardised (send to Bo Stig Hansen, bsh@id.dth.dk). We are hopeful that this work will be able to take place under the auspices of the BSI, in the same way as Z has recently been "adopted" without yet formally entering standardisation,
We would welcome participation in this activity, at any level, by anyone interested in RAISE. Please contact me for further details.

Formal Methods Tools Database

Tim Denvir
timdenvir@cix.compulink.co.uk

With the blessings of FME (Formal Methods Europe) and BCS FACS (Formal Aspects of Computing Science) I am initiating a database of information on formal methods tools. Details of the database will available to all members of FACS and of FME via periodic announcements in the FACS Europe newsletter. Under normal conditions suppliers of tools will pay a small fee to enter and maintain details of their tools on the database, but readers of the database will have *free* access either through FACS Europe or via ftp which is being arranged from several sources. The readership of FACS Europe number about 700, a highly focussed list of computer scientists interested in formal methods.

I am offering to those who demonstrate their tools at either the FACS Refinement Workshops or at the FME Symposia a year's free entry of details of their tools on to the database. Those who demonstrated their tool at FME'93 or the VIth Refinement Workshop will be eligible for free entry during 1994 and those who demonstrate at FME'94 will be eligible for free entry during 1995.

This database will therefore be a service to FACS and FME members and an additional free service to demonstrators at FME and Refinement Workshops. For information on how to enter details of a tool, the template to fill in etc. please e-mail me at the address below.

I shall require that in the database, descriptions of tools should be without hype, comparisons with other tools or claims which are not objectively determinable. This means that the use of words like "good", "excellent", "mature", "industrial quality" should be avoided. I reserve the right to edit entries in the description, but if I do so, the final version will be submitted to the tool supplier for agreement before entering it in the database. I hope that it will not often be necessary to do this. The choice of information in the record has been guided by the appendix in the proceedings of FME'93, LNCS 670.

I am still trying to determine what the normal terms should be for suppliers, who have not demonstrated at FME or Refinement Workshops; I wish to find a price which leaves me not burdened with a large amount of unpaid work and yet which tool suppliers will feel is value for money. Some general principles are: each renewal of the entry of a tool carries an opportunity to update the details; the fees are on a per tool basis; suppliers will not be able to renew for more than one year at a time; and updating details will always incur a fee, even for those who started with free entry. First-time demonstrators at FME or Refinement Workshops will have a year's free entry for that tool into the database; demonstrators whose tools are already resident will be granted a waiving of six months' fees. The database will be updated with all the updates received and paid for to date every three months; thus any update will be accessible within three months of receipt and payment.

The above terms are tentative and I need to work on them in more detail; but they express my intention of how the database will be operated. When it has all been operating for some time I may think of extending the operation to include independent assessments, but I plan to start with simple beginnings.

Meanwhile, please propose a demonstration of your tool to FME'94; if you demonstrate there, provided there are no unforeseen circumstances and the scheme is running, you will be entitled to a free year's entry (six months if your tool already has free entry resulting from demonstrating at a previous event).

At present there are 24 tools in the database summarised overleaf.

Formal Methods Tools Database

Name of Tool	Supported	Contact Name	email
B-Toolkit	Abstract Machine Notation	Ib Holm Sorensen	Ib.Sorensen@comlab.ox.ac.uk
Boyer-Moore theorem prover	Boyer-Moore logic	William D. Young	young@cli.com
CADiZ	Z core language and mathematical toolkit	David Jordan	yse@minster.york.ac.uk
Centaur	generic	Janet Bertot	jmi@sophia.inria.fr
Centaur-VDM environment	VDM-SL	Phillippe Facon	facon@cnam.cnam.fr
Design/CPN	Coloured Petri Nets	John Moelgaard	jm@elctr.dk
DisCo	DisCo language	Kari Systa	ks@cs.tut.fi
DST-fuzz	Z	Hans-Martin Horcher	
ExSpect	Hierarchical coloured timed Petri Nets	L.J.A.M. Somers	wsinlou@win.tue.nl
FDR	CSP	David Jackson	David.Jackson@prg.ox.ac.uk
Formaliser	Z	Susan Stepney	susan@logcam.co.uk
ForMooZ MooZ	(Modular Object-Oriented Z)	Silvio Lemos Meira	srlm@di.ufpe.br
IFAD VDM-SL Toolbox	BSI-VDM	Poul Boegh Lassen	poul@ifad.dk
IPTES Toolset	SA/RT, SA/SD	Rene Elmstrom	rene@ifad.dk
LOTOS Toolbox	LOTOS	A.W. van der Vloedt	vdvloedt@ita.nl
Mathias	Prolog (various systems)	Dr. Ron Knott	R.Knott@surrey.ac.uk
Mural	VDM-SL (subset)	Dr. Brian Ritchie	br@inf.rl.ac.uk
Oyster Whelk CLaM Barnacle	Martin-Lof Type Theory, Prolog, Goedel, Lazy ML	Prof. Alan Bundy, Dr. Geraint A. Wiggins	geraint@ai.ed.ac.uk
Pet Dingo	Estelle	Brett W. Strausser	strauss@osi.ncsl.nist.gov
ProofPower	HOL, Z	Roger B. Jones	R.B.Jones@uknet.ac.uk
PVS		Dr. Natarajan Shankar	shankar@csl.sri.com
RAISE	RSL	RAISE	raise@csd.cri.dk
SpecBox	VDM-SL	Peter Froome	pkdf@dcs.ed.ac.uk
TAV	CCS	Kim G. Larsen, Arne Skou	{kgl,ask}@iesd.auc.dk

At this stage please contact the tool suppliers direct to obtain more details of their tool.

Recent books column

Cliff Jones

February 4, 1994

I agreed to produce listings of books which relate to the purpose of this newsletter. Authors should send references (BibTeX format preferred) to `cbj@cs.man.ac.uk`. For this edition, I have gone back to 1990 and obtained input by putting out a request on `comp.specification`.

Authored books: [MP92, Mid93, Mor90, BA90, EM90, Hen90, Daw91, LBC90, AI91, WH93, Odi90, BW90, JJLM91, Jon90, BFL+94, Gro92, Mos92, Ost90, Fra92, WSL93, Wor92, Spi92, vdS93, Old91, AO91, BA93, PST91, TZ88, Win93, DS90, Dil90, GS93, Inc92, Mey90, Fei93]

Edited book: [LH94a, Yon90, Par90, Bae90, JS90, FvGGM90, GH93, MW93, Ros94, Coh90, Hay93, LH94b, GM93, Mel93]

Proceedings: [Ame91, IM91, Nie93, BG91, PT91a, PT91b, BJ90, GJ93, WL93, Bes93, BHL90]

References

[AI91] D. Andrews and D. Ince. *Practical Formal Methods with VDM*. McGraw-Hill, 1991.

[Ame91] P. America, editor. *ECOOP'91*, volume 612 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[AO91] Krzysztof R Apt and Ernst-Rüdiger Olderog. *Verification of Sequential and Concurrent Programs*. Springer-Verlag, 1991. ISBN 0-387-97532-2, 3-540-97532-2.

[BA90] M. Ben-Ari. *Principles of Concurrent and Distributed Programming*. Prentice Hall, 1990. ISBN 0-13-711821-X.

[BA93] M Ben-Ari. *Mathematical Logic for Computer Science*. Prentice Hall International, 1993. ISBN 0-13-564139-X.

[Bae90] J. C. M. Baeten, editor. *Applications of Process Algebra*. Cambridge University Press, 1990.

[Bes93] E. Best, editor. *CONCUR'93: 4th International Conference on Concurrency Theory*, volume 715 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[BFL+94] J. C. Bicarregui, J. S. Fitzgerald, P. A. Lindsay, R. Moore, and B. Ritchie. *Proof in VDM: A Practitioner's Guide*. FACIT. Springer-Verlag, 1994. ISBN 3-540-19813-X.

[BG91] J. C. M. Baeten and J. F. Groote, editors. *CONCUR'91 – Proceedings of the 2nd International Conference on Concurrency Theory*, volume 527 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[BHL90] D. Bjørner, C. A. R. Hoare, and H. Langmaack, editors. *VDM'90: VDM and Z – Formal Methods in Software Development*, volume 428 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[BJ90] M. Broy and C. B. Jones, editors. *Programming Concepts and Methods*. North-Holland, 1990.

[BW90] J. C. M. Baeten and W. P. Weijland, editors. *Process Algebra*. Cambridge University Press, 1990.

[Coh90] Edward Cohen. *Programming in the 1990's, An Introduction to the Calculation of Programs*. Texts and Monographs in Computer Science. Springer-Verlag, 1990. ISBN 3-540-97382-6.

[Daw91] J. Dawes. *The VDM-SL Reference Guide*. Pitman, 1991.

[Dil90] Antoni Diller. *Z - An Introduction to Formal Methods*. John Wiley and Sons, 1990. ISBN 0-471-92489-X.

[DS90] Edsger W Dijkstra and Carel S Scholten. *Predicated Calculus and Program Semantics*. Springer-Verlag, 1990. ISBN 0-387-96957-8, 3-540-96957-8.

[EM90] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1990.

[Fei93] Loe Feijs. *A Formalisation of Design Methods - a Lambda-calculus approach to system design with an application to text editing.* Ellis Horwood, 1993. ISBN 0-13-106113-5.

[Fra92] Nissim Francez. *Program Verification.* Addison Wesley, 1992. ISBN 0-201-41608-5.

[FvGGM90] W H J Feijen, A J M van Gasteren, D Gries, and J Misra, editors. *Beauty is our Business - A Birthday Salute to Edsger W Dijkstra.* Springer-Verlag, 1990. ISBN 0-387-97299-4, 3-540-97299-4.

[GH93] John V. Guttag and James J. Horning. *Larch: Languages and Tools for Formal Specification.* Texts and Monographs in Computer Science. Springer-Verlag, 1993. ISBN 0-387-94006-5/ISBN 3-540-94006-5.

[GJ93] M-C. Gaudel and J-P. Jouannaud, editors. *TAPSOFT'93: Theory and Practice of Software Development*, volume 668 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[GM93] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993. ISBN 0-521-44189-7.

[Gro92] The RAISE Language Group. *The RAISE Specification Language.* BCS Practitioner Series. Prentice Hall, 1992. ISBN 0-13-752833-7.

[GS93] David Gries and Fred B Schneider. *A Logical Approach to Discrete Math.* Springer-Verlag, 1993. ISBN 0-387-94115-0, 3-540-94115-0.

[Hay93] Ian Hayes, editor. *Specification Case Studies.* Prentice Hall International, second edition, 1993.

[Hen90] M. Hennessy. *The Semantics of Programming Languages.* John Wiley, 1990.

[IM91] T. Ito and A. R. Meyer, editors. *TACS'91 – Proceedings of the International Conference on Theoretical Aspects of Computer Science, Sendai, Japan*, volume 526 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[Inc92] D C Ince. *An Introduction to Discrete Mathematics, Formal System Specification and Z.* Oxford University Press, 1992. ISBN -19-853836-7.

[JJLM91] C. B. Jones, K. D. Jones, P. A. Lindsay, and R. Moore. *mural: A Formal Development Support System.* Springer-Verlag, 1991. ISBN 3-540-19651-X.

[Jon90] C. B. Jones. *Systematic Software Development using VDM.* Prentice Hall International, second edition, 1990. ISBN 0-13-880733-7.

[JS90] C. B. Jones and R. C. F. Shaw, editors. *Case Studies in Systematic Software Development.* Prentice Hall International, 1990. ISBN 0-13-116088-5.

[LBC90] J. T. Latham, V. J. Bush, and I. D. Cottam. *The Programming Process: An Introduction Using VDM and Pascal.* Addison-Wesley, 1990.

[LH94a] Kevin Lano and Howard Haughton, editors. *Object-Oriented Specification Case Studies.* Prentice Hall, 1994. ISBN 0-13-097015-8.

[LH94b] Kevin Lano and Howard Haughton, editors. *Object-oriented Specification Case Studies.* Prentice Hall International, 1994. ISBN 0-13-097015-8.

[Mel93] T. Melham, editor. *Higher Order Logic and Hardware Verification.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1993. ISBN 0-521-41718-X.

[Mey90] Bertrand Meyer. *Introduction to the Theory of Programming Languages.* Prentice Hall International, 1990. ISBN 0-13-498502-8.

[Mid93] Cornelius A. Middelburg. *Logic and Specification: Extending VDM-SL for advanced formal specification.* Chapman and Hall, 1993.

[Mor90] Carroll Morgan. *Programming from Specifications.* Prentice-Hall, 1990.

[Mos92] Peter D. Mosses. *Action Semantics.* Number 26 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, 1992.

[MW93] Ursula Martin and Jeannette M. Wing, editors. *First International Workshop on Larch, Dedham 1992.* Workshops in Computing. Springer-Verlag, 1993. ISBN 3-540-19804-0, 0-387-19804-0.

[Nie93] Oscar M. Nierstrasz, editor. *ECOOP'93: Object-Oriented Programming*, volume 707 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Odi90] P. Odifreddi, editor. *Logic and Computer Science*. Academic Press, 1990.

[Old91] E-R Olderog. *Nets, Terms and Formulas*. Cambridge University Press, 1991. ISBN 0-521-40044-9.

[Ost90] Jonathan S. Ostroff. *Temporal Logic for Real-Time Systems*. Advanced Software Development. Reserach Studies Press (distributed by John Wiley & Sons, 1990. ISBN 0 471 92402 4.

[Par90] H. A. Partsch. *Specification and Transformation of Programs: A Formal Approach to Software Development*. Springer-Verlag, 1990.

[PST91] Ben Potter, Jane Sinclair, and David Till. *An Introduction to Formal Specification and Z*. Prentice Hall International, 1991.

[PT91a] S. Prehn and W. J. Toetenel, editors. *VDM'91 – Formal Software Development Methods. Proceedings of the 4th International Symposium of VDM Europe, Noordwijkerhout, The Netherlands, October 1991. Vol.1: Conference Contributions*, volume 551 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[PT91b] S. Prehn and W. J. Toetenel, editors. *VDM'91 – Formal Software Development Methods. Proceedings of the 4th International Symposium of VDM Europe, Noordwijkerhout, The Netherlands, October 1991, Vol.2: Tutorials*, volume 552 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[Ros94] A. W. Roscoe, editor. *A Classical Mind: Essays in Honour of C. A. R. Hoare*. Prentice-Hall, 1994.

[Spi92] J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, second edition, 1992.

[TZ88] J V Tucker and J I Zucker. *Program Correctness over Abstract Data Types, with Error-State Semantics*. North-Holland, 1988. ISBN 0-444-70340-3.

[vdS93] Jan L A van de Snepscheut. *What Computing is all About*. Springer Verlag, 1993. ISBN 0-387-94021-9, 3-540-94021-9.

[WH93] M. Woodman and B. Heal. *Introduction to VDM*. McGraw-Hill, 1993.

[Win93] Glynn Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, 1993. ISBN 0-262-23169-7.

[WL93] J. C. P. Woodcock and P. G. Larsen, editors. *FME'93: Industrial-Strength Formal Methods*, volume 670 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Wor92] J. B. Wordsworth. *Software Development with Z*. Addison-Wesley, 1992.

[WSL93] M. Weber, M. Simons, and Ch. Lafontaine. *The Generic Development Language Deva: Presentation and Case Studies*, volume 738 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993. ISBN 3-540-57335-6.

[Yon90] Akinori Yonezawa, editor. *ABCL: An Object-Oriented Concurrent System*. MIT Press, 1990. ISBN 0-262-24029-7.

The VDM Forum

A new e-mail list for researchers, practitioners and teachers

A new e-mail list has been set up for discussions on any aspect of system specification and development using the Vienna Development Method (VDM). Anyone with an interest in VDM (research, teaching or industrial) is welcome to subscribe to the list.

The group is informal, friendly and wide-ranging. Topics for discussion will include, but are not limited to:

Problems, tips and techniques of specification and refinement;
Information and discussion about tools;
Conference, course, workshop and seminar announcements;
Reports on academic and industrial projects;
VDM-SL semantics and proof theory;
The VDM-SL Standard.

Associated with the list will be a repository of files from which members of the list can obtain copies of public documents including The VDM Bibliography, the Draft Standard, project reports and the discussion list archives.

To subscribe, send an e-mail message to

mailbase@mailbase.ac.uk

containing the following line alone as the message body:

join vdm-forum Joe Bloggs

where Joe Bloggs is your name (two words only; complex names must be hyphenated, e.g. Joe van-der-Bloggs). The subject line is immaterial. Your e-mail address will be picked up automatically. You will be sent an information file on how to post etc. on joining the list.

The vdm-forum list is unmoderated, but is administered on a voluntary basis by John Fitzgerald at Newcastle University (vdm-forum-request@mailbase.ac.uk).

The vdm-forum list is provided by courtesy of the UK Network Information Services Project and JANET. The list may not be used for commercial gain.

Formal Computer Science in JFIT

Dan Simpson
University of Brighton
ds33@unix.btom.ac.uk

This year the JFIT Annual Report comprises six volumes: an overall report is supported by detailed reports on the programme areas in VLSI technology, advanced devices and materials, systems architectures, communications and distributed systems, and systems engineering. Together these reports form the best overview available of government funded research in the United Kingdom. They are available (free) from TPS1a, DTI, 151 Buckingham Palace Road, London SW1W 9SS, England.

The overview volume contains a number of reports from the various divisions of JFIT, including education and training. There are also detailed lists which include the number of research students allocated to various institutions and the courses which SERC supports.

Here we shall concentrate on the systems engineering work and more particularly the sub-programme within that area called formal computer science. Overall within the systems engineering area there were 300 grant applications in the academic-only programmes with a requested value of £23m.

The report states a concern that, although the quality remains high, only a small proportion of these - approximately 17% - will eventually be funded.

The figures for the FCS area in the year 92/93 are as follows. Total applications 68 with value £9,690k. Of these, 17 alpha rated projects were funded with a value of £2,410k but 37 alpha rated projects remain unfunded with a value of £5,490k. There were six beta rated projects and 8 rejected.

This year saw the end of the Logic for IT initiative, which has been a major success in this area. For further details see EATCS Bulletins 40 and 51. The community still waits with baited breath whether a follow-on programme will be announced.

Within the area of collaborative programmes probably the one which was most relevant to readers was the safety critical systems programme. The programme has been reviewed with the conclusion that the workplan has now been covered. The SERC funding has been fully committed and almost all the DTI funding was committed. From the two calls 52 proposals were received; 15 first call and 21 second call projects have been approved and are now running. The programme involves some 80 industrial collaborators from a large cross-section of UK industry.

Within the overall systems engineering area there are 159 grants held by HEIs with a total value of £21,530k; for the formal computer science area there are 66 grants with a value of £9,772k. As can be readily seen, FCS is by far the most popular area both in number of grants and in value.

Within the FCS programme there are 36 universities and 32 companies involved. The universities with the number of projects and £k of grant are listed overleaf.

All the information in this section is abstracted from the JFIT Annual Report. Being Government statistics they may not be too accurate! A very simple analysis shows a number of inconsistencies but at a broad level they give a picture of what is going on. For more details you should obtain copies of the reports from DTI at the address given above.

Of the 66 FCS grants the software engineering volume of the annual report contains one page summaries of the following projects. Presumably the other grants did not have reports ready for the publication date.

Formal Computer Science Projects

D Sannella - Edinburgh Computer Assisted Formal Reasoning: Formal Development of Programs from Specifications

J W Lloyd - Bristol Foundations of Meta-Programming in Logic Programming

D A Turner - Kent Machine Supported Verification of Functional Programs

M Fourman - Edinburgh Formally Based Systems Design Tools

C P Stirling - Edinburgh Modal and Temporal Mu-Calculi

S B Cooper - Leeds Partial Functions, Non Deterministic Computations and Polynomial Time Enumeration Reducibility

J K Truss - Leeds Equality in Logic Programming Word Problems and Unification

C A R Hoare - Oxford The Formal Design of Medical Diagnostic Computer Programs

J V Tucker - Swansea Logic Programming, Abstract Data Types and Many-Sorted Model Theory

D Rydeheard - Manchester Programming Environments and Categorical Logic (II)

A J Sinclair - Edinburgh Quantitative Analysis of Stochastic Systems in Computer Science

A J R G Milner - Edinburgh Declarative Languages and Applied Semantics

R Kennaway - East Anglia Generalised Graph Rewriting as a General Computational Model

C M Holt - Newcastle Embedding Concurrent and Imperative Programming Constructs in Interval Temporal Logic

T Rodden - Lancaster Database Requirements for Co-operative Working

M Levene - University College, London Development of Software Engineering Database System Based on Hypernode Model.

D M Gabbay - Imperial Syntactical Foundations of Non-monotonic Reasoning

A M Pitts - Cambridge Verifying ML Programs using Evaluation Logic

R Burstall - Edinburgh Constructive Logic as a Basis for Program Development

K Bennett - Durham A Proof Theory for Program Refinement and Equivalence

G D Plotkin - Edinburgh Logical and Semantical Frameworks

M M Y Croft - Cambridge Types, Strictness Analysis and Reduction Machines

R G Wilson - Warwick Symmetry Breaking in Neural Networks for Visual Pattern Recognition

C P Stirling - Edinburgh Verification of Concurrent Infinite State Systems

D Simpson - Brighton Developing and Using Formal Models of Inheritance

R Cooper - Glasgow Configurable Data Models

H Barringer - Manchester Model-checking Unbounded State Space Programs

I Hodkinson - Imperial Theory of Interval Time Handling

M Shanahan - Imperial Logic of Knowledge Representation

D Sannella - Edinburgh Algebraic and Logical Foundations of Formal Software Development

J Darlington - Imperial Definitional Constraint Programming: A Foundation for logically Correct Concurrent Systems

A G Cohn - Leeds Logical and Computational Aspects of Spatial Reasoning

A G Cohn - Leeds Declarative Extensions of Logic Programming

I A Stewart - Swansea Descriptive Complexity Theory

C M N Tofts - Swansea Process Semantics for Simulation and its Applications

D Sannella - Edinburgh Formal Development of Modular Programs in Extended ML

D Sannella - Edinburgh Formal Development of Modular Programs for Algebraic Specifications: Computing Support

M Thomas - Glasgow Further Verification Techniques for LOTOS Specifications

C A R Hoare - Oxford Probably Correct Hardware/Software Co-design

K J Turner - Stirling FORMOSA (Formalisation of Open Systems Architecture)

Projects and Grants

Bath (1 - 230)	Lancaster (1 - 243)
Birkbeck (1 - 99)	Loughborough(3 - 249)
Birmingham (1 - 275)	Manchester (2 - 380)
Brighton (1 - 109)	Newcastle (2 - 184)
Cambridge (3 - 264)	Open (1 - 86)
City (2 - 506)	Oxford (1 - 97)
Cranfield (1 - 150)	Paisley (1 - 304)
Durham (1 - 94)	QMWC (1 - 92)
East Anglia (1 - 25)	Royal Holloway(3 - 451)
Edinburgh (10 - 1962)	St Andrews (1 - 70)
Essex (1 - 54)	Stirling (1 - 61)
Exeter (1 - 388)	Surrey (1 - 110)
Glasgow (3 - 84)	Sussex (1 - 126)
Heriot Watt (2 - 517)	Swansea (1 - 96)
Hertfordshire (1 - 55)	Ulster (1 - 41)
Imperial (3 - 467)	UMIST (1 - 84)
Keele (1 - 102)	Warwick (2 - 201)
Kent (1 - 164)	York (7 - 1403)

Report on Formal Methods Tutorial

Angela Alapide
Aerospace Systems Division
Space Software Italia
alapide@ssi.it

Space Software Italia (SSI) is a company based in Taranto, Italy, specialized in the design and implementation of aerospace systems, which has identified as strategic for future exploitation the participation in LaCoS, an ESPRIT project aimed at demonstrating the applicability of Formal Methods to the industrial development of software. SSI has been recently in charge of disseminating theoretical information on Formal Methods and of reporting about industrial experiences related to their use in practise, within the companies controlled by Alenia, its mother company.

Among other activities performed in this frame, SSI gave a half-day tutorial to managers and engineers coming from the Alenia Corporate Divisions interested in the Formal Methods technology. The tutorial set out the following objectives:

- Introduce Formal Methods from a methodological perspective in terms of: definitions; indications on the ways in which they can be used for the production of high quality software; kind of support they can provide to the various phases of the software lifecycle.
- Give to the attendees an appreciation of what a formal method looks like. The formal method presented was RAISE. The presentation focused much on the RAISE Specification Language (RSL) than on the method. Specifically the tutorial included a large description of RSL constructs and of the various specification styles supported by RSL.
- Point out the pre-requisites and the problems, together with possible solutions, related to the integration of Formal Methods in the existing software development environments.

The tutorial was attended by 20 people, newcomers to Formal Methods, who expressed interest on the subject "Formal Methods". They participated actively to the tutorial by submitting to the presenters many questions on the subjects treated and, moreover, raising further issues. Some concerns were expressed by the Quality Assurance responsibles regarding: the need of establishing a sort of mapping between products associated to traditional development processes and their corresponding (if any) when a formal approach is applied; the changes in the effort distribution required by the adoption of Formal Methods. Other problems pointed out by most attendees were related to the technology transfer process and the need of having Formal Methods integrated with the everyday software development practices. Nevertheless, no cultural prevention was perceived in the attendees and, moreover, people involved in the development of safety-critical software, feel Formal Methods are techniques whose adoption will be in the mid-long term more and more required by their customers.

Report of the BCS FACS Sixth Refinement Workshop 5 - 7 January 1994

The Sixth Refinement Workshop was held at City University on 5-7 January 1994. It follows five previous workshops held at approximate annual intervals; the last was in January 1992, again in London at Lloyd's Register of Shipping.

The sixth workshop was plagued with curses: two of the invited speakers were unable to come at the last minute owing to unforeseeable circumstances, the session on the second day was interrupted by a (genuine) fire alarm for over an hour, and on the third day David Till, the joint workshop chairman and local organiser, slipped on the very icy pavements and broke his wrist. Despite these troubles, the workshop was a great success: the standard of the papers and their delivery was exceptionally high and the audience, at just over 50 in number from ten different countries, smaller than in previous years, participated actively with lively questions and discussion.

David Garlan's invited paper, "Using Refinement to Understand Architectural Connection", explored the use of refinement to specify and re-use the connectors between components in a system's architecture. Common architectural ideas such as client-server ports and connections were illustrated as interacting protocols defined in a subset of CSP.

Session 1 comprised three papers and the first of several opportunities to see tool demonstrations. The first paper was by Kevin Lano and Howard Haughton, "Improving the Process of System Specification and Refinement in B". They saw two barriers to the widespread adoption of formal methods in the development of high integrity systems: that there are few methods for integrating formal methods with current practice such as ERA models, especially useful for formalising the early stages of the life-cycle, and the difficulty of performing proofs in practice, especially of refinements. The paper demonstrated how ERA models, with inheritance and specialisation, can be expressed in the Abstract Machine language of the B method. Refinement to code was illustrated and the proof obligations identified. The formalisation of a dynamic model was shown using aspects of a lift system as an example.

The "Formal development of Authentication Protocols" by Pierre Bieber and Nora Boulahia-Cuppens again used B to define a specification of a protocol with various security aspects involving malicious agents. Cryptographic keys are used to ensure the required security properties and refinement properties shown. The authors used the B-tool to build and verify the various specifications and refinements in the paper.

"Testing and Safety Analysis of AM Specifications" by Howard Haughton and Kevin Lano again used Jean-Raymond Abrial's Abstract Machine language (AM) as a formal medium for specifications. A thesis of the paper is that safety analysis is related to testing. A tree can be constructed in which the top node denotes the conjectured fault and the lower nodes denote causes for the fault. By using substitution axioms establishing post-conditions for AM expressions, a means of selecting test cases is shown. This is then related to a fault tree analysis in a safety analysis context.

The second invited speaker, Willem-Paul de Roever, was unable to be present owing, we understand, to ill-health. His paper, co-authored by a team of four, was most ably delivered by Jan Peleska. I think he deserves a special mention because, not only did he step into the breach at short notice, but he succeeded in holding the audience's interest with a lucid talk despite being interrupted by an evacuation of the workshop following a fire alarm. The paper, "Formal Semantics for Ward and Mellor's Transformation Schema", addressed the divide between a structured analysis and design method and formal approaches; this is considered an important issue at the present time, as it seems to be a way of moving formal approaches further towards the requirements analysis end of the life-cycle. Ward-Mellor's method is claimed to be used by 1/6 of all system specifiers in the USA but, the authors state, is in places inconsistent or ambiguous and does not provide any characterisation of real-time behaviour with enough rigour to express or deduce specific timing properties. However the method contains enough indications to enable attempts at reconstructing its intended meaning. The paper identifies a number of these ambiguities, explores the plausible intended meanings and proposes some semantics. These are presented as labelled transitions in the style of Plotkin.

In session 2 Lindsay Groves presented a case study in combining program specialisation and data refinement, "Deriving Language recognition Algorithms". His paper showed very eloquently how several language recognition algorithms could be derived from a single abstract algorithm. He derived a LR(1) parser via specialisation and refinement, deferring data refinement until later in the development process.

The second paper in session 2 was by J. von Wright: "Program Refinement by Theorem Prover". He showed how HOL and the "window inference system" can be used to prove refinements, according to Back's refinement calculus. The window inference system is a tool developed by Jim Grundy (and reported in the fifth Refinement Workshop) which supports a transformational style of reasoning with HOL. The window inference tool enables refinements of expressions to be deduced from refinements of sub-expressions ("windows"). Weakest precondition semantics presented as a formal system, the resulting predicate transformation system can be embedded into HOL. The predicate transformations must have certain "healthiness" conditions, e.g. monotonicity with respect to statement composition and predicate conjunction, certain continuity conditions etc. The chosen predicate transformations may be implementable, but not necessarily so: they could be more general and include, for example, angelic nondeterminism. Data refinement is achieved through abstraction and (inverse) representation relations applied locally over windows.

The first paper in session 3 was "Co-refinement" by Mike Ainsworth and Peter Wallis. This treats of the situation where one has two or more specifications with possibly different frames and/or signatures implicit in their pre and post-conditions, each specification representing a different "viewpoint" of a user or designer of part of the system. The specifications can be combined using a number of different combinators, including union which if it is applied in a state satisfying both pre-conditions, will guarantee that both post-conditions are satisfied. Co-refinement is a partial ordering relation between specifications which is equivalent to refinement if the signatures of the specifications are the same, and which has useful properties when related to the various forms of specification combination.

The second paper of session 3 was by Raymond Nickson and Lindsay Groves: Metavariables and Conditional Refinements in the Refinement Calculus. This describes two techniques for the refinement calculus which facilitate goal-directed development. Decisions about the precise form of refinement steps can be deferred, so that high-level choices can be expressed as soon as they are appropriate. Metavariables are place-holders for components of partly developed programs which will be instantiated when they are suitably constrained by later refinements. Conditional refinements allows the development of alternative refinements into a guarded command set. A rigorous way of applying these techniques was described and illustrated.

The last paper in session 3 was "Machine Code Programs are Predicates too" by Theodore Norvell. This considered two kinds of computational behaviour: that specified by high-level programs in terms of source level variables and that specified by machine language programs in terms of registers and memory. The two are related by using predicate logic as a common framework. The relationship serves as a specification for a code generator. The idea clearly has relevance for provably correct compiler generation.

The third and final invited paper was from Steve Schuman and David Pitt on "Object-Oriented Formal Specification and Behavioural Refinement". The example cnosen to illustrate the principles was a lift system. A Z-style of notation was used to express the state of the system and also its behavioural properties in terms of state-transitions associated with events. Further extensions and elaborations were explored.

The fourth session started with a paper from Xu Qiwen and He Jifeng. They explore algebraic laws of parallel programming with shared variables. The programming language chosen was a variant of that of Owicki and Gries. Non-deterministic choice plays an important part in the refinement calculus of statements in this language, being equivalent to a g.l.b. operator w.r.t. the refinement ordering. For me, this put into a new context the non-deterministic aspects of Dijkstra's guarded commands; although they do not involve parallelism, if one equips them with a refinement order, then a non-deterministic operator such as Hoare's union operator completes the non-deterministic aspects and, I think, turns the refinement order into a lattice and provides the set of statement forms with a pleasing symmetry. This may be an obvious insight, but because of it I found this paper particularly rewarding.

The second paper in session 4 was from Yves Ledru and Pierre Collette: Environment-based Development of Reactive Systems. They approach the specification of reactive systems with the environment as the starting point of the development. Lamport's TLA framework is used and the canonical case study of the dining philosophers shows how the method prevents over-specification. The environment provides a context for proving the design and validation activities.

In the fifth and last session, Kevin Lano's paper: "Refinement in Object Oriented specification Languages" addressed the semantics and refinement of object oriented Z. He argued that a clear semantic framework can be developed based upon the standard Z specification language which can include simple forms of temporal reasoning and the treatment of object identity. It supports global and local reasoning about the properties of specifications.

The second paper of the session was "Operation Semantics with Read and Write Frames" by Juan Bicarregui. This paper explored in detail the semantic models of the "external" clauses in VDM specifications, which bind the free variables which appear in operation pre and post-conditions and also indicate the sets of state variables which implementations are allowed to read and write. The consequences are more subtle than most people have realised and the author proposed an extension to the denotational model of operations which captures this informal understanding of the read frame. The exploration uses notions of satisfiability (or equivalently, refinement) and was illustrated very usefully with specific cases and diagrams.

The final paper was "Proof obligations for Real-Time Refinement" by Colin Fidge. This extends existing algorithm design rules for refining Z specifications to structured high-level implementations with proof obligations which preserve specified real time behaviour in addition to the normal functional behaviour. A linear time model is used, specified in Z, and related to refinement in the context of real time requirements, as may be found in some safety applications.

Seven tools were demonstrated. Contacts for most of these are provided elsewhere in this newsletter, in the article about the Formal Methods Tools Database. The tools demonstrated were: CADIZ, FDR, Formaliser, Matthias, Oyster/Whelk/CLaM/Barnacle, RAISE, Z.

City University provided an efficient and amenable environment for the workshop with working facilities for the demonstrations, a very pleasant workshop dinner, a book display by Springer in whose "workshops in computing" series we expect to publish the proceedings. FACS warmly thanks City university, the organisers and the contributors for a most successful event.

Finally, I apologise in advance for any misconceptions or misrepresentations. Any errors are mine and any differences in detail of reporting should be attributed to my inconsistent concentration during the event.

Tim Denvir

Report of the 1993 BCS-FACS Christmas Meeting

Formal Aspects of Object Oriented Systems

**was held at the Department of Computing,
Imperial College of Science Technology and Medicine (London)
on December 16th and 17th 1993.**

The aims of the meeting were to review recent work on the logical basis of Object-Oriented structure, formal support for Object-Oriented system development, the application of Object-Oriented structuring to the development of large scale specifications and formal treatment of concurrency in Object-Oriented systems. It was well attended and we were ably hosted by Imperial College. The standard of presentation was very high with speakers introducing some parallel processing with the formal and explanatory threads of the talks concurrently presented.

SESSION 1

The first session (on December 16th) was principally concerned with the application of Category theory to specification structuring and the first talk was by Jose Fiadeiro (University of Lisbon) and Tom Maibaum (Imperial College): *Fundamentals of Object Oriented Structuring* and integrated two paradigms: the temporal logic imperative and the categorical imperative. Thus, object specifications are temporal theories and communities (systems of interconnected component objects) are diagrams. The notion of an object specification was defined and applied to a producer-buffer-consumer example. Accordingly buffers are defined as objects with associated actions *get(ITEM)* and *put(ITEM)* and axioms which describe the object including temporal operator *next*. This object (together with objects consumer, buffer and communication channels) is a node of a diagram whose edges are signature morphisms. Joint behaviour of this system is given by the *colimit* of the diagram; taking the colimit computes the disjoint union of the signatures and identifies the symbols which are shared between the components. For instance, *put* (from the buffer) and *store* (from the producer) are identified as a single action of the system. A second illustrative example was presented which provided a seasonal flavour : a variation on the Dining Philosophers problem, with the philosophers transformed to Santa's and the forks transmuted to reindeer reins.

The second talk was given by Grant Malcolm (Oxford University): *Equational Specification of Systems of Interacting Objects* and this was a study of ways of specifying systems of interacting objects and ways of composing specifications to reflect the concrete composition of systems. In order to model objects with local states the notion of *Hidden Sorted Algebra* was developed, and this was described in the talk. *Sheaves* were used to formalise the passage from local to global variables and the pasting together of local observations of behaviour; a system is defined as a diagram of sheaves connected by sheaf morphisms. The example given to illustrate the talk was a system consisting of two automata where the output of one automaton is the input of another.

Frank Piessens (speaker) and Eric Steegmans (Catholic University, Louvain) followed with: *Categorical Semantics for Object-Oriented Data Specifications.* In this semantics nodes and arrows of a graph respectively represent classes and dependencies between classes, and this was illustrated by means of a specification of a simple library system where classes are *Person*, *Book* etc. A special form of specification is defined, viz, a *canonical form*, and a proof was presented that no two non-isomorphic canonical specifications have the same semantics.

SESSION 2

This principally concerned the formal development of object oriented programs and the first speaker was Cliff Jones (University of Manchester) who presented his work on: *A Concurrent Object-based Design Notation: Concurrency and its Semantics.* The talk was chiefly about the design notation $\pi o\beta\lambda$ which is employed in the development of O-O concurrent programs. $\pi o\beta\lambda$ is heavily based on the POOL programming language; in addition there exists a mapping between $\pi o\beta\lambda$ and the π-calculus. $\pi o\beta\lambda$ supports composition in parallel programming by making it possible to avoid interference. An illustrative example was given of a sorted list (class) with methods *add* (a member to the list) and *remove* (the smallest member of the list). The semantics of $\pi o\beta\lambda$ is such that only one method can be active in any one instance of a class at a time. The code which invokes a method is held in a rendezvous until the method being executed reaches a *return* statement. A sequential version of the class was first provided, with *return* as the last statement contained in each method. It was then transformed to a more efficient parallel form, so that *add* and remove contain *return* as their first statement; as soon as the parameter is passed the caller is released from the rendezvous. The rule for this interchange is that *S; return e* can be replaced by *return e; S* provided that *S* always terminates, *e* is not affected by *S* and *S* only invokes methods reachable by private reference. Thus concurrency is allowed by making sure the returns are executed as soon as possible.

Paulo Borba (Oxford University) presented the next talk on: *An Operational Semantics for FOOPS. FOOPS* (Functional and Object-Oriented Programming System) is a functional, concurrent, object-oriented specification language with an executable subset and derived from OBJ. A structural operational semantics was provided for FOOPS and the notion of refinement between specifications defined in terms of the semantics. A stack provided an illustrative case study to demonstrate the process of formal software development in FOOPS.

SESSION 3

This session chiefly concerned the integration of formal and structured methods and began with a talk by Chris Dollin (Hewlett-Packard) on: *The Rationale Behind the Fusion OOA/D Method. Fusion* is a method developed by HP which originated from a training course in object-oriented programming for HP engineers. The method involves the designer first building three interacting models, an object model, an operational model and a lifecycle model. (This is in order for an understanding of the system to be developed.) The object model incorporates classes and relationships, viz an E-R diagram; the operation model includes pre- and post-conditions and the lifecycle model specifies accepted sequences of events. The subsequent design activity implements the system operations via interacting objects and implementation follows in which the design is converted into object-oriented programme code. The speaker made the point that although Fusion is not a formal method (which require powerful tools which are currently unavailable) it does have formal underpinnings. For example the pre- and post-conditions in the operational model are declarative in nature and can be written formally if desired. Also the lifecycle model is based on regular expressions and admit of straightforward implementation; altogether the methods can be regarded as the "Trojan Horse" approach to formality! There are several tools available to support Fusion (eg FUSIONCASE), but none developed by HP.

Kevin Lano (of Lloyds Register) presented the next talk on: *Integrating Formal and Structured Methods in Object-Oriented System Development*, co-authored by H. Houghton and P Wheeler. The speaker described the formalisation of models expressed in the *Object Modelling Technique* (OMT) notation by Rumbaugh. Systematic mapping techniques were described which capture the standard meaning of data and dynamic models of the OMT notations in a formal specification in either Z or the B Abstract Machine Notation. A (safety-critical) case study was described: a system which optimises and monitors the loading of bulk carriers to ensure that safe hull stress limits are not exceeded during loading, and to create optimised loading plans.

SESSION 4

This session began bright and early at 9.00am (on December 17th) with Steve Schuman and David Pitt (University of Surrey) who combined to give a presentation on: *Object Oriented Formal Specifications* and *the Rest Stays Unchanged: State Based Concurrency.* Their main concern was to be able to reason about dynamic (behavioural) properties of some specified class of object . This is based on an ability to reason about static (structural) properties of the formal specification itself. The first part of the talk defined events in terms of post-conditions characterising its effect as a possible change of state. The specifications used a Z-like notation. Thus an event is expressed as a relation between pre-states and their possible post-states, both of which must also satisfy the state-invariant. The speaker then established some formal validation conditions for such specifications. Eg *There is some state in which the event can start* and *the event may successfully complete from every state from which it can occur.* The specifications were then refined and an observation made that refinement and composition are just logical conjunction ie adding/combining constraints. The formal validation conditions serve to ensure that there are at least some models for the resulting specification. *Putting specifications together* to make specifications is regarded as the real pay-off of Object-Orientation. This minimalist approach to formal specification was further discussed in the second part of the talk, by David Pitt on: *The Rest Stays Unchanged.*

David Pitt gave several examples to illustrate the relational view of pre/post conditions where there are parts of the state being changed which are unaltered by the event. This more readily allows the creation of complex specifications from simpler constituents since the constituents only specify in an explicit way parts of the state. Neutral relations were defined, which keep part of the state fixed, allowing the rest to (possibly) alter. In conclusion, if events have duration we are able to reason about concurrent behaviour by considering the overlapping of two events in terms of conjunction.

The controversial subject of inheritance was tackled by Jim Armstrong (DCSC, University of Newcastle) who spoke on: *The Impact of Inheritance on Software Structure.* (The talk was co-authored with the MFI Group, University of Brighton.) He began by reviewing previous formal perspectives (of inheritance) which have tended to be operational and denotational semantic models. In particular Cook and Palsberg have concluded that inheritance provides "expressive power not available in other languages". Examples were provided of the use of inheritance, such as representing IS_A and PART_OF relationships. However there have also been warnings that inheritance mechanisms are potentially harmful and an opposing view of inheritance is taken by Magnusson who has compared it to the "goto" statement! The speaker's view was that inheritance hierarchies require formal guarantees that valid type inclusion relations are properly captured and this could be achieved by formulating models of both inheritance and module hierarchies. A translation function is then defined from the former to the latter. In conclusion there is an increasing likelihood that real-time languages such as Ada 9X with inheritance capabilities will be employed in the production of safety-critical systems where formality is vital.

SESSION 5

This session opened with Eugene Durr and Stephen Goldsack presentation of their work on the *Afrodite* project: *Approaches to Specifying Real-Time Requirements and Concurrent Behaviour in VDM++*. Stephen Goldsack (Imperial College) described the specification language VDM++, an object-oriented extension of VDM. This offers objects derived from classes with inheritance and polymorphism, supporting reuse of specifications. Refinement (reification) is supported by object structures to represent VDM structures such as maps and sequences. In addition, the scope of VDM is extended to include concurrency and real time applications. Independent (concurrent) execution threads are modelled by active objects (process objects) and concurrency achieved by the invocation of passive objects by these threads. One means of enforcing synchronisation discipline is the use of permission predicates derived from deontic logics (Model Action Logic). The talk was illustrated by a re-visit to our well known Dining Philosophers; the philosopher represented an active and the table a passives class.

Eugene Durr (Utrecht University) continued with an exposition of some real time (RT) principles, with RT considered as (in some sense) orthogonal to the usual functional specification. In order to model continuous variables (such as temperature, current etc) these quantities are sampled at intervals. In addition methods have associated time durations. An electronic circuit was given as an example, comprising resistance, inductance, capacitor, source. Each of these are modelled as subclasses of component and the whole circuit then modelled by a world of parallel objects: resistance, inductance etc, each having their own independent thread. Together they form a (discrete) model of the circuit differential equation.

Sophia Drossopolou (Imperial College) presented the last talk of the session which was co-authored by Stephan Karathanos and entitled: *Static Typing for Dynamic Binding. ST&T* is a new type system for Smalltalk which includes the following feature: the subtype relationship is an extension of the subclass relationship and a method may have several signatures. As a consequence type-checking a method does not require re-type-checking the same method in the sub-classes of the class that contains the method body. In addition more programs are type correct using ST&T than in previous type systems. In order to cope with the possibility of multiple signatures the type of message expression (or message sent) is determined by those signatures of the message which “fit best” the types of the receiver and arguments; inference rules were presented for determining the types for message expressions.

SESSION 6

Keith Clark (Imperial College) presented: *Distributed Symbolic OO Programming Using April. April* (Agent Process Interaction Language) is a language for implementing distributed symbolic applications. Application areas include distributed AI and multi-agent systems and April will shortly be used in a project for British Telecom on distributed fault finding over a network. It is a process language where processes communicate Prolog style recursive data structures to mailboxes using TCP/IP; objects can be emulated as processes, as in concurrent logic programming and new processes can be spawned either locally or remotely. Other features include the use of guarded commands (similar to CSP) and the use of functional expressions, a departure from traditional logic programming. Additionally April can communicate with other Unix processes via a suite of routines for reading and writing to mailboxes as though they were files; examples are DIALOX, Motif.

The session concluded with a further talk with a distributed application theme, by Yiping Yang and Nicolas Treves (Telesystemes, Guyancourt, France): *Introducing OO Concepts into a Net-Based Hierarchical Software Development Process.* The talk was based on the *PROOFS* project work on the promotion of the use of net-based techniques in the development of Heterogeneous Distributed Applications. *Channel/Agency* (CA) nets were used to for modelling purposes in the Requirements Analysis and Systems Architecture phases of the system lifecycle. A CA net consists of a set of labelled active (agency) and passive (channel) components with appropriate links. O-CA nets are an extended version of CA nets which include some O-O concepts and characteristics wherein an object is represented by an agency and operations among objects by channels. For example if object A uses object B then the are both linked to the same *uses* channel. Refinement is accomplished by replacing certain objects by linked object-uses-object groupings; inheritance is captured by embedding sub-nets into nets. Net development is achieved by an iterative use of refinement, inheritance and splitting of *use* channels. The completed O-CA net is linked to Place transition and Coloured Petri Nets for the latter stages of system design and subsequent code implementation is in C++ or Ada. This development process was illustrated by a Document Conferencing Application where Editors desk and Producers desk objects are both linked to Document object via separate *uses* channels. This top-level net was then developed to become a system comprising an editor, many producers and a document containing many parts.

SESSION 7

The last session of the meeting was opened by Emil Sekerinski (Universitat Karlsruhe) who introduced: *Refinement Algebra for Object-Oriented programming.* A formal treatment of the concepts of encapsulation, instantiation was given based on the refinement algebra (Back, Morgan). Each object is defined by means of a signature which maps method names to value and result parameter names and types. A class (of objects with a particular signature) is subsequently refined to produce a class with the same behaviour. For example a buffer defined more generally as a bag of items is refined by a FIFO buffer. Relations between object types (such as refinement, specialisation etc.) are formally defined and proof theoretic criteria are derived (based on simulation) for verifying these relations. By these means an object-oriented refinement algebra is developed which is powerful enough for the study of many aspects of object-oriented programming and is able to reason about programs and specifications equally well.

Ana Moreira presented: *LOTOS in the Object-Oriented analysis process* (by A Moreira and R.G. Clarke of the University of Stirling). The ROOA *(Rigorous Object-Oriented Analysis)* method combines O-O methods and formal description techniques to produce a formal O-O analysis model that acts as the requirements specification of a system. The formal model is expressed in LOTOS and, as it is executable, prototyping tools can be used to help validate the specification against the original requirements. For this to be possible, a LOTOS interpretation is given of O-O constructs: for example aggregation is modelled by defining a process for the aggregate class which embeds a process for each component together with an interface process.

The last talk was by Ian Maung (University of Brighton): *Behavioural Subtyping and Substitutability.* The speaker compared inheritance with the "goto" statement and predicted that inheritance would be eventually be replaced by abstraction. Thus the objectives of his work were to formalise a general inheritance mechanism, distinguishing different uses of inheritance. An object-oriented programming language (OOPL) is being designed supporting abstractions only. He illustrated by formalising the IS_A relation by putting together its behavioural and contravariant aspects. This was in order to provide objective criteria for checking its correctness. The talk ended with an outline of an OOPL syntax and the identification of some remaining problems such as the need for a proof theory for subtypes.

Margaret West,
University of Leeds

Summary of the ERIL Project

This project, officially called *Verification Techniques for LOTOS Specifications*, but otherwise known as the *ERIL project* (for *Equational Reasoning in LOTOS* and the ERIL software tool) was funded by the joint SERC/DTI Information Engineering Directorate (IED) programme. It involved four partners for the following periods:

University of St Andrews *Lead Partner*
Prof. U. Martin (October 1989-August 1993)

University of Glasgow
Dr. M. Thomas (October 1989-September 1992)

British Telecom PLC
Dr. E. Cusack (October 1989-July 1990)

Rutherford Appleton Laboratory
Dr. B. Ritchie (October 1989-March 1993)

The aim of the project was to investigate the verification requirements of LOTOS [ISO:8807] specifications, and to determine the applicability of equational reasoning and term rewriting to discharging those requirements.

This work was presented at major international conferences throughout the life of the project and has resulted in about 11 high quality journal publications, 25 publications in the proceedings of major international conferences, and a further 25 technical reports.

In addition two significant pieces of software were developed: an ASN.1/LOTOS translator; and the ERIL equational reasoning system.

Further work continues in several of these areas under SERC GR/J31230 (Prof. Martin), GR/J08300 (Dr. Thomas) and GR/J52716 (Dr. Thomas).

The main scientific achievements of this project were:

- **Case studies** The completion of major case studies in safety-critical application areas in collaboration with researchers from end user organisations including:
  - A medical information bus, in collaboration with Royal Free Hospital School of Medicine.
  - A control device for a radiation machine, carried out in collaboration with DEC/SRC Palo Alto.
  - A secure login protocol in collaboration with a major defence contractor.
  - GKS (Graphical Kernel System), in collaboration with numerous end-users.

  These studies extended the use of LOTOS, commonly thought of as appropriate only for specifying protocols, by showing how it could be used in a variety of safety critical applications. They also showed the importance of formal specifications for uncovering errors.

- **Verification requirements** A clear understanding of the diverse verification requirements that different applications of LOTOS may generate, gained from both theoretical analysis and investigation of case studies.

  Our results are very much in line with the conclusions of a recent wide ranging American DoD study of formal methods. The simplest verification requirements are often the most important to users, and any methodology should give some guidance as to what to do if the verification requirements are not satisfied.

- **Discharge of verification requirements** A clear understanding of the most appropriate way to use tools in the discharge of these requirements. Tools should be simple and provide automated support of common decision procedures, together with useful guidance when proofs fail.

  In particular the Larch Prover, was used in the discharge of requirements in three of the main studies described above, and proved ideal for the rapid development and debugging of proofs. Our own ERIL system incorporates experimental techniques devised more specifically to discharge the verification requirements.

- **Reasoning techniques**

  Significant original research on foundational issues:

  - The development of new techniques for equational reasoning incorporated in the ERIL prover and other systems,
  - termination, and
  - divergence.

The main technical achievements of the project comprised:

- **The ASN.1 to LOTOS translator**

  The ASN.1 to LOTOS translator is a software tool for translating data type specifications written in the language ASN.1 into LOTOS.

  This type of project proved to be a successful application area for formal methods: formalisation of the language revealed several inconsistencies and omissions from the language design, and functional programming as a prototyping tool enabled quick and effective communication between language designers, implementors, and users.

  The translator is not only the first formally specified tool for ASN.1, but it is also the first translator for the complete language.

  The tool is in use at British Telecom and currently discussions are under way with three companies (in Austria, Canada and UK) with a view to further exploitation.

- **The ERIL prover**

  ERIL is an equational reasoning theorem prover based on first order term-rewriting. It is highly reconfigurable with an advanced user interface. It supports order sorted logic, which allows the succinct representation of many complex problems, and is the only such prover to support specialised inference mechanisms for order-sorted logic.

  The system was one of the eight major equational reasoning systems chosen for demonstration at the Sixth International Conference on Rewriting Techniques and Applications (RTA) 1993, in Montreal.

For further information about the project please contact:

Prof. Ursula Martin
Department of Mathematical
and Computational Sciences,
University of St Andrews
St Andrews
Scotland
Phone: 0334 63252
E-mail
`um@dcs.st-and.ac.uk`

or

Dr M Thomas
Department of Computing Science
University of Glasgow
Glasgow G12 8QQ
Phone: 041 330 4969
E-mail `muffy@dcs.glasgow.ac.uk`

A Comparison of the Conventional and Formal Design of a Secure System Component

T.M.Brookes, M.A.Green[*] J.S.Fitzgerald[†] P.G.Larsen[‡]

February 18, 1994

British Aerospace (Systems and Equipment) Ltd, (BASE) is developing a security critical device, a Trusted Gateway. Some preliminary work on the application of formal methods to its design was performed as a case study by the University of Newcastle . A proposal for an application experiment based on this programme was submitted to and approved by the European Systems and Software Initiative (ESSI). This programme is being used to assess the use of formal methods in the system and software design process within BASE. This article describes the basic philosophy, objectives and methodology of the ESSI experiment which is now getting underway at BASE Plymouth.

1 Introduction

British Aerospace (Systems and Equipment) Ltd. (BASE) currently produces several security-critical systems. At present, the use of formal methods in the design of these systems is not mandated but, as the confidence in the security-related features has to be increased, their use is desirable. This confidence is generated by being able to show that all of the requirements have been captured and analysed thoroughly and that traceability from requirement to implementation has been established.

A programme is currently underway within BASE to develop a Trusted Gateway (Network Guard) which has limited functionality, making it ideal for testing formal methods against current design practices. The evaluation level sought for this device requires that some aspects of the design be modelled formally, even though the design does not *have* to be carried out using formal methods.

Some initial work was carried out on the Trusted Gateway with the assistance of the University of Newcastle. From an initial specification developed within BASE, a formal specification covering many aspects of the gateways operation was developed. This early work has been used to formulate the specification to be used in the ESSI work.

The Trusted Gateway programme is wholly funded by BASE and the EC. The University of Newcastle will act in the role of consultant to the project where their expertise and familiarity

[*]British Aerospace (Systems and Equipment) PLC, Clittaford Road, Southway, Plymouth, Devon, PL6 6DE (email: mikeg@cx.plym.ac.uk, tel: +44 752 695695, fax: +44 752 695500)

[†]Department of Computing Science, University of Newcastle upon Tyne, Newcastle upon Tyne NE1 7RU, UK (email: John Fitzgerald@newcastle.ac.uk, tel: +44 91 222 8058, fax: +44 91 222 8788

[‡]IFAD, "The Institute of Applied Computer Science", Forskerparken 10, DK-5230, Odense M, Denmark (email: peter@ifad.dk, tel: +45 65 93 23 00, fax: +45 65 93 29 99)

with formal methods will prove valuable. IFAD will supply a tool set to support the VDM formal method and will also act as consultants in its use. The results obtained in this programme will be reported at the ESSI Workshops and to other meetings.

2 Trusted Gateways

The Trusted Gateway being developed in this experiment is a simple device consisting of a single input and two outputs. Messages are read into the Gateway, analysed for the presence of character strings which indicate the message classification and then written to the appropriate (high or low security) output port. In an information processing system, the Trusted Gateway would be placed in the communications path between two systems which have different security levels. Its purpose is to sort the information into different classifications and ensure that it is sent to the correct destination. This is shown diagramatically in Figure 1.

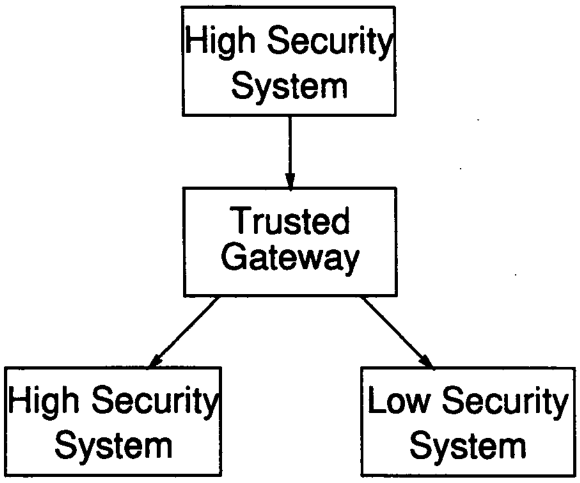

Figure 1: System Function of the Trusted Gateway.

There are many exception conditions which have to be catered for in the design. For example:

- what does the system do with any characters which are read in which do not form a part of a message?,
- what happens if the message is longer than a pre-defined number of characters ?

These conditions have to be addressed and system solutions provided for them. Formal methods provide techniques for modelling the system at various levels of abstraction which can help to detect the presence of these events and to check that the proposed design deals with them correctly.

3 Objectives of the Application Experiment

The primary objective is to determine whether the system design process from requirements capture to software development and test can be improved by introducing formal methods. Parameters used to make this judgement include the effort expended in the design and evaluation processes and the number of customer requirements which are satisfied on the first design iteration. Other aims are to investigate if the use of formal methods:

- produces a more reliable product,
- decreases the time to respond to changes in the customer's requirements, and
- produces a product which better satisfies the customers' requirements.

The experiment should show the level of cost effectiveness of developing a secure system using a formal design techniques in the context of an existing design methodology.

The subsiduary objective of the experiment is to define where in the project life cycle formal methods generate the most benefit and whether the entire design has to be analysed or can the majority of the benefit be obtained by only considering a part of the entire system. If sucessfull, the parts of the design process in which the use of formal methods give the most benefit will be identified.

4 The Proposed Methodology

The experiment consists of the parallel development of the Trusted Gateway by two separate and independent design teams. Procedures, such as ensuring that the design teams are in different areas and are instructed not to discuss the project with each other, will be enforced to minimise the communication between them so that they are truly independent. The first team will use the standard BASE development methodology, Ward and Mellor[WM85], supported by the Teamwork Computer Aided System/Software Engineering tool set[1]. The other team will follow a similar design process, but will use formal methods to support this design methodology and to develop a formal specification. The specification will be developed and tested using the VDM tool kit from IFAD which allows the specification to be animated. Test cases can be developed and applied to the specification to investigate whether the results are as expected. These test cases can then be used in the test of the final product. Rigerous proof of the formal specification will not be attempted during this experiment.

The experiment will take the design processes from the Customer Requirement through System Design to Development and Implementation. There will be a central authority acting as the customer and keeping records of the queries, problems and progress made on the project. In the system design stage, a change in the specification will be introduced to obtain a measure of the effort required to implement the change and the effect on the design process. The team members will not know in advance what change will be made to the specification. At the conclusion of each stage of the process, a design review will be performed on each specification. The customer will examine the output and compare it with what was required. No feedback will be performed, but the deficiencies will be recorded in the experiment design log.

Each version of the system will be implemented in software and tested using the test plan devised by its own design team. The test procedures from the other design team will then be applied to compare the test coverage achieved by the two design approaches. The software produced will be tested by the original customer to determine if it fulfils his requirements. Areas where the performance is deficient will be highlighted and examination of the records of the project will pinpoint the decisions in the design process responsible for the introduction of that

[1]Teamwork is a Registered Trademark of Cadre Technology Inc.

area. Actions will then be taken to introduce additional controls to the existing procedures to reduce the probability of a similar event occurring in future projects

The following points will be assessed and used to compare the two design process:

1 The effort required to perform the design task (It is realised that those using formal methods will be inexperienced. Provision has been made to support them in the method, but not the design detail, using the external consultants: University of Newcastle and IFAD.);

2 The number and complexity of the queries raised against the requirement specification;

3 The number and seriousness of the deficiencies identified during the design reviews;

4 The number of identified inconsistencies detected within the customer requirement;

5 A comparison of the compliance of the system the original specification;

6 After implementation, the performance of the system and tests in terms of code size, speed, accuracy and test coverage will be made.

The reliability of both products will be evaluated, based on the rate of detection of problems throughout the development, testing and evaluation periods.

The proposed experiment is shown in outline in Figure 2. It consists of three parallel activities, each of which is split into 4 major phases. The first activity is the BASE conventional project described above. The second activity is a duplicate of the first in which the formal specification language VDM-SL[2] is used to support the design, development and test process. The third activity monitors, compares and reports on the progress in each of the design activities. The monitoring activity includes the review and reporting of progress on the application experiment.

Support will be available to the engineers during the course of the programme. Consultants will be available to advise on both the method (University of Newcastle and IFAD) and the use of the tools set (IFAD) to try to compensate for the lack of on-site expertise in the use of formal methods.

Training in the use of formal methods and the toolset will be provided for the engineers involved in the design and monitoring process. The tools set provides a means for preparing VDM specifications in the correct format, performs static checking, and can animate the VDM-SL specification produced during the course of the design [LL91]. IFAD and Newcastle will also participate in the review of the programmes where the design methods are compared. Their comments on the progress of the experiment will be incorporated in the reports.

The experiment will be initiated by producing a customer requirement that is submitted to each of the independent design teams. Queries on the requirement and the clarification of the requirement will be submitted and responded to in writing. Requests submitted by each team will not be passed on to the other. The number of requests and their complexity will be used as a measure of how well the requirement capture process employed forces the system designer to clarify the requirement with the customer. There is likely to be a divergence in the design as each of the groups will interpret the original specification differently and will receive answers to different questions. After the system design has been produced, a major change will be made

[2]The project will use VDM in conformance with the ISO/BSI Draft Standard [ISO93]

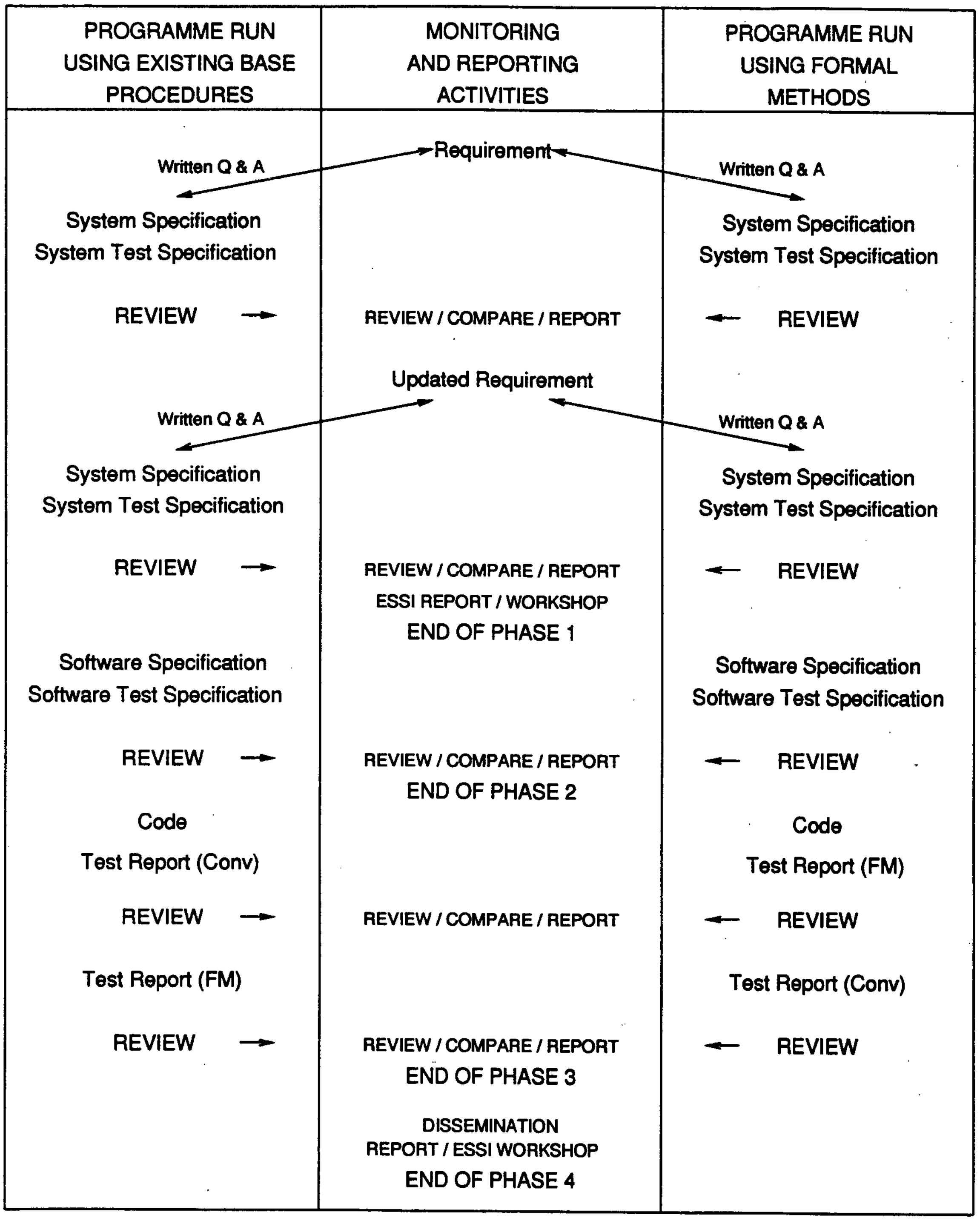

Figure 2: An Outline of the Programme.

in the customer requirement (the design teams will not be expecting it) and the effort required to include the change will be measured.

4.1 Phase One

During Phase One, BASE will produce the documentation required for the design process. The major documents produced are the System Specification and the System Test Plan that is used as the final customer acceptance document. The equivalent documents will be generated in the parallel (formal methods) path. After each document has been produced it will be subjected to review and then updated if necessary. After this review, a major change will be made to the customer requirement and new issues of the specification will be generated to reflect this change. The documents will then be reviewed.

The final activity of the first phase is a major review in which the progress and results of the two design teams will be contrasted and compared. The output of the review will be a document showing the progress to date and comparing the two approaches.

4.2 Phase Two

The second phase commences with the submission of the system documentation to the software department for design. Two new independent design teams will be employed, one using the conventional specification and one the formal specification. The customer in this case is represented by the system engineer who produced the system documentation. Queries and responses on the documents are to be provided in writing to enable the understanding of the requirements to be measured and compared in the two teams.

A subset of the required BASE documentation, sufficient for software implementation, will be produced during the second phase. The primary documents to be produced are a Software Specification and Test Plan. The phase concludes by reviewing and comparing the two approaches and reporting the results.

4.3 Phase Three

In the third phase, the software design documentation will be used to produce prototype code. A third team of engineers will perform this task using the software design documents as their source of information. The special purpose hardware required for a secure implementation will not be available, so the code will be run on a suitable computer. The implementations will be tested using their own Test Specifications and deficiencies in the code corrected. The coverage of the Test Plans and the detection of faults in the implementations will be compared. As a final test, the Test Plans for each development path will be applied to the other implementation to compare how each implementation satisfies the customer's requirements. The original customer will also examine the implementations to determine how well they satisfy the original requirement.

4.4 Phase Four

In the final phase of the project, the resuts from the project are analysed and the two approaches compared. The final report on the programme will be produced which will describe any benefits which have been obtained using a formal design approach.

5 Data Capture and Analysis

The Customer Requirements and the System Security Policy have been captured using the RTM requirement traceability tool[3]. As queries are received, they will be recorded against the requirement and clarification text will be added to amplify the requirement when needed. This will be used as the basis for requirement 'expansion Using the 'trace' facility, the evolution of the final system requirement from the initial customer specification can easily be followed.

6 Project Status

The project commenced at the start of 1994. No results have been obtained yet. The conclusion of Phase One is expected in April 1994, Phase 2 concludes during September and Phase Three terminates in November. Final results from the experiment are expected early in 1995.

7 Conclusions

In this article, we have described the basic function of a Trusted Gateway and discussed our experiment in using formal specifications at various stages of the system development. It should be stressed that we are not employing a full formal method. Rather, we are experimenting with the application of formal specification in the context of an existing design methodology (which has been certified to ISO 9000). The application of formal techniques is modest: it is not envisiged that refinment obligations or other proofs will be conducted by the engineers (although the consultants may perform some proofs in private). The checking of the formal specification will be performed using the specification animation capability of the IFAD VDM toolbox.

The results of the experiment will show whether using formal methods can be improves the system and software design and that they are applicable in an industrial environment. The results of this experiment will influence whether BASE adopts a formal design apprach for future projects or parts of projects.

The authors welcome comments and expresssions of interest from other systems developers, especially in security-critical fields.

8 Acknowledgements

The authors acknowledge the support of the British Aerospace Dependable Computing Systems Centre at the Universities of Newcastle upon Tyne and York, which sponsored the initial study on which the ESSI experiment was based.

References

[ISO93] ISO. Document Number ISO/IEC JTC1/SC22/WG19/N-20 Information Technology Programming Languages - - VDM-SL First Committee Draft Standard CD 13817-1, November 1993.

[3]RTM is a Registered Trademark of GEC Marconi, Addlestone, Surrey

[LL91] P. G. Larsen and P. B. Lassen. An Executable Subset of Meta-IV with Loose Specification. In *VDM '91 - Formal Software Development Methods*. Springer-Verlag, October 1991.

[WM85] P.T. Ward and S.J. Mellor. *Structured Development for Real Time Systems*, volume 1,2,3. Yourdon Press, Prentice-Hall, Englewood Cliffs, NJ, 1985.

EATCS

Dan Simpson, University of Brighton

This note is to introduce EATCS to readers of the newsletter. Whilst many readers may already be members of EATCS, with the expanded circulation of the newsletter some people may not know of EATCS or its activities.

EATCS is the European Association for Theoretical Computer Science. It is a loose association of people in industry and academe who are interested in both developing the foundations of our subject and also applying them in order to build better software and computer based systems. The aims are very similar to those of both FACS and FM Europe but, just as FACS and FM Europe have their own distinctive but complementary flavours so too has EATCS.

To many EATCS members the most important reason for joining is the EATCS Bulletin. This is a large compendium of news, reports, technical papers, conference announcements and general information to keep people up to date in the area. Personally I consider it the only really essential reading I receive. It is published three times a year, about each academic term, by a group in Leiden and is edited by Gregorz Rozenberg who is also the current president of EATCS. It is worth joining the association if only to obtain the Bulletin.

But EATCS has many other activities of its own and also helps promote many joint activities. On the publications front, as well as the Bulletin there is also a journal - Theoretical Computer Science; a series of books and monographs published by Springer Verlag; and various occasional publications such as conference proceedings. All these are offered at a discount rate to members. The association also agrees deals with various other publishers so that many publications can be obtained at a discount rate.

The association is also active in the conference area where the most important activity is ICALP the association's annual conference.

Despite the title and the firm European base of the association it has ties with national groups in Europe and also around the world. So if you wish to keep up to date with activities in places such as Japan, Australia and the USA the regular news columns in the Bulletin are a must. There are also regular news columns from various European organisations including BCS-FACS.

I can only assume that readers of this newsletter must be interested in what is going on in our subject and I fully recommend that you help meet this interest by joining EATCS. Further details are available from the contact address given but you can easily join EATCS via FACS. Contact the FACS secretariat at Loughborough and they will send you a form which will allow you to invest £10 in a way which will repay itself many times each year.

Contact
Prof Dr B Monien
Secretary EATCS
Fachbereich Mathematik - Informatik
Universitat-GH Paderborn
33098 Paderborn
Germany
Email eatcs@uni-paderborn.de

DUES

The dues are DM 30.– for a period of one year. If the initial membership payment is received in the period December 21 – April 20, April 21 – August 20 or August 21 – December 20, then the first membership year will start on June 1, October 1 or February 1, respectively. Every contribution payment continues the membership for the same time period. Payments can be made in DM and in US $.
Additional fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe. The amounts are DM 12.– for USA, Canada and Israel, DM 16.– for Japan and DM 20.– for Australia per year. For information on additional fees for other destinations contact either the Secretary or the Treasurer.

HOW TO JOIN EATCS

To join, send the annual dues, or a multiple thereof (to cover a number of years), to the Treasurer of EATCS:

Prof.Dr. D. Janssens
University of Antwerp (UIA)
Dept. of Mathematics and Computer Science
Universiteitsplein, 1
B-2610 Wilrijk, Belgium
Email: dmjans@wins.uia.ac.be
Fax: (+32) 3 820 2421
The dues can be paid (in order of preference) by cheques in DM or US $, cheques in other currency, DM or US $ cash, or other currency cash. When submitting your payment, please make sure to indicate your complete name and address. For this purpose you may use the form below.
Transfers of larger amounts may be made via the following bank account, on the condition that an additional DM 8.– is paid for each transfer to cover the bank charges. If you use this way of payment, please send the necessary information (reason for the payment, name and address) to the treasurer.
Our account is:
Generale Bank Antwerpen
Antwerp (Wilrijk), Belgium
Account number: 220-0596350-30-01130

I would like to join EATCS / renew my EATCS membership and enclose .... as membership fee

for ...... years (and .... for air mail delivery)

Name:

First name:

Address:

Date: Signature:

BCS FACS
"A Brief History of Formal Methods"
Bernie Cohen

Thursday 28 April 1994

5.30 - 7.00 p.m.

Room M405, the Middleton building, City University

167-173 Goswell Road, London EC1

Bernie Cohen has a long experience in promoting formal approaches, both in industry and academia, and is known to many of us for his enthusiastic commitment to formality in software engineering. This talk is, in his words, "a very personal view, anecdotal rather than encyclopaedic, but accurate as far as it goes".

FREE for members of FACS, £5 for others.

Please register your intention to come by contacting:

BCS-FACS
e-mail: facs@lut.ac.uk
fax: 0509-610815
tel: 0509-222676
c/o Department of Computer Studies,
University of Technology,
Loughborough, LE11 3TU.
preferably by e-mail

Call for Participation

8th Z User Meeting – ZUM'94

Organized by the Z User Group in association with BCS FACS
Sponsored by BT, Logica Cambridge Limited & Praxis
Supported by the ESPRIT ProCoS-WG Working Group (no. 8694)

St. John's College, University of Cambridge, England

Tutorials:	**27–28 June, 1994**
Main meeting:	**29–30 June, 1994**
Educational Issues:	**1 July, 1994**

Programme committee:

ZUM'94, the 8th Z User Meeting, is to be held at St. John's College, University of Cambridge, on 29th and 30th June 1994, preceded by two days of tutorials covering introductory Z, object-oriented Z specification, B, the B-method and B-toolkit, project management with formal methods, and real-time system development.

An Educational Issues session on 1st July provides a forum for educators and others to discuss issues relating to the teaching of formal methods, in general, and Z, in particular.

The meeting will also include tool demonstrations and displays by publishers. A small display of computing memorabilia is also planned.

For further details and general enquiries about the meeting contact:

Jonathan Bowen (Conference Chair, ZUM'94)
Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK.
Tel: +44-865-283512 (direct), 283521 (secretary)
Fax: +44-865-273839 or 273819
Email: `Jonathan.Bowen@comlab.ox.ac.uk`

or

Neville Dean (Educational Issues, ZUM'94)
Anglia Polytechnic University
Cambridge CB4 4ES, UK.
Tel: +44-223-63271
Fax: +44-223-352979
Email: `cdean@va.anglia.ac.uk`

Guidelines for Newsletter Contributions

Contributions may be in the form of **single-sided** camera-ready copy, suitable for layout and sub-editing. They can also be sent to us using electronic media (i.e. by floppy disk (MS DOS or Mac)/e-mail/etc.), to be formatted in the house style. As a rule, we generally accept pure ASCII text or TEX/LATEX in order to avoid complications involving interchange between wordprocessing formats. We regret that we are unable to offer typesetting facilities for handwritten material.

If contributions are sent using proprietary wordprocessor/markup language formats (i.e. MicroSoft Word 5, FrameMaker), then these will be treated as though they were camera-ready copy. If we are unable to print them adequately or to otherwise convert to another more suitable form then the authors may be asked to provide paper copies of appropriate reproduction quality.

Artwork can be provided for appropriate inclusion, either using general formats (such as DVI files or Encapsulated PostScript) by sending camera-ready paper copy. Generally, line drawings and other high-contrast graphical diagrams will be acceptable.

Material must be of adequate quality for reproduction. Output from high quality printers with at least 300 DPI resolution is generally acceptable. Output from printers with lesser resolution (i.e. dot-matrix printers) tends not to reproduce very well and will not be of sufficiently good print quality. The Editorial Panel reserves the right to refuse publication for contributions which cannot be reproduced adequately.

Page definition information

If possible, contributions should be designed to fit standard A4 paper size, leaving a margin of at least one inch (1") on all sides. Camera ready copy should be sent in **single-sided** format, with page numbers written lightly **on the back**. Ideally, all fount sizes used should be no smaller than 10pt for clarity. Contributions should attempt to make adequate use of the space, filling at least 60% of each page, including the last one. Authors should note that all contributions may be sub-edited appropriately to make efficient use of space.

Deadlines

The production deadlines for the coming year are:

Summer	end of May, 1993	**Winter**	end of November, 1993
Autumn	end of August, 1993	**Spring**	end of February, 1994

Disclaimer

The views and opinions expressed within articles included in the FACS Europe newsletter are the responsibility of the authors concerned and do not necessarily represent the opinions or views of the editorial panel.

Addresses

Editors:
Dr. Jawed Siddiqi
Dept. of Computing and Management Sciences
Sheffield Hallam University
100 Napier Street
Sheffield, S11 8HD
United Kingdom

Tel: +44 742 533141
E-mail: `J.I.Siddiqi@shu.ac.uk`

Dr. Brian Monahan
Dept. of Computer Science
University of Manchester
Oxford Road
Manchester, M13 9PL
United Kingdom

Tel: +44 61 275 6137
E-mail: `brianm@cs.man.ac.uk`

BCS FACS and FME Committee 93-94

General

General enquiries about the BCS FACS group, the newsletter or its meetings can be made to:

BCS FACS
Department of Computer Studies
Loughborough University of Technology
Loughborough, Leicestershire
LE11 3TU
Tel: +44 509 222676
Fax: +44 509 211586
E-mail: FACS@lut.ac.uk

Membership fees 1994

Standard (i.e. non-BCS members)	:	£25
BCS members	:	£10

Discount subscription rates 1994

EATCS	:	£10
FACS Journal	:	£35 (6 issues, Vol. 6)

FACS Officers

Chair	Tim Denvir
Treasurer	Roger Stone
Committee Secretary	Richard Mitchell
Membership Secretary	John Cooke
Newsletter Editors	Jawed Siddiqi & Brian Monahan
Publicity	Brian Monahan
Liaison with FACS Journal	John Cooke
Liaison with BCS FMIS group	Ann Wrightson

FACS Committee Members

Name	Affiliation	Tel:	E-mail
D. Blyth	Incord Ltd.	0202-896834	DBlyth@cix.compulink.co.uk
J. Boarder	Buckinghamshire	0494-22141	jcb@buckscol.ac.uk
R.E. Carsley	Westminster	071-911-5000x3568	roger@westminster.ac.uk
D.J. Cooke	Loughborough	0509-222676	D.J.Cooke@lut.ac.uk
B.T. Denvir	Translimina Ltd.	081-882-5853	timdenvir@cix.compulink.co.uk
S.J. Goldsack	Imperial	071-589-5111x5014	sig@doc.ic.ac.uk
A.J.J. Dick	Bull	0442-884586	J.Dick@brno.uk03.bull.co.uk
R.B. Jones	ICL Winnersh	0734-693131x6536	
R.J. Mitchell	Brighton	0273-642458	rjm4@unix.brighton.ac.uk
B.Q. Monahan	Manchester	061-275-6137	brianm@cs.man.ac.uk
M.P.Naftalin	Lloyd's Register	081-681-4040	tcsmpn@aie.lreg.co.uk
J.I.A. Siddiqi	Sheffield Hallam	0742-533141	J.I.Siddiqi@shu.ac.uk
D. Simpson	Brighton	0273-600900x2450	ds33@unix.bton.ac.uk
R.G. Stone	Loughborough	0509-222686	R.G.Stone@lut.ac.uk
D.R. Till	City	071-477-8552	till@cs.city.ac.uk
M.M. West	Leeds	0532-335430	mmwest@scs.leeds.ac.uk
A. Wrightson	Central Lancashire	0772-893242	annw@sc.uclan.ac.uk

FME Officers

		Name	Affiliation	Tel:	E-mail
Chair	:	Martyn Thomas	Praxis plc	+44-225-444700	mct@praxis.co.uk
Secretary	:	Tim Denvir	Translimina Ltd.	+44-81-882-5853	timdenvir@cix.compulink.co.uk
Treasurer	:	Kees Pronk	T.U.Delft	+31-157-81803	c.pronk@twi.tudelft.nl