FACS FACTS

The Newsletter of the BCS Formal Aspects of Computing Science SIG

Series II Vol 2 No 2 Aug '91.

Editor Jawed Siddiqi

The summer newsletter contains lots of goodies. We have included reports of two of our meetings provided by our roving reporter Tim Denvir. First, the B-tutorial held at Manchester University PEVE unit which focused on the B proof system. Second, the formal aspects of measurement workshop, for which the reports and abstracts are included.

Readers of FACS FACTS will be pleased to know that because of my heroic achievemnts in compiling the newsletter I have been honoured with the distinguished offer of obtaining an extremely limited edition of "The Collected Internal Memoranda of F. X. Reid".

You will see this month that we have included the ICL Z Proof Tool Newsletter. We are always looking for technical reports, summaries of research activities, particularly from Industry.

Happy Summer

Jawed Siddiqi

Chairman:
Dr. D. John Cooke
Department of Computer Studies
Loughborough University of Technology
LOUGHBOROUGH
Leicestershire
LE11 3TU
Tel: (0509) 222676
FAX: (0509) 610815
E-mail: D.J.Cooke@uk.ac.lut

Secretary:
B. Tim Denvir
Praxis Systems Plc
20, Manvers Street
BATH
BA1 1PX
Tel: (0225) 444700
FAX: (0225) 465205
E-mail: btd@uk.co.praxis

(Currently seconded to:

DTI/ITD Room 815
Kingsgate House
66-74, Victoria Street
LONDON
SW1E 6SW
Tel: (071) 215 2353
E-mail: btd@uk.co.praxis
)

Treasurer:
Dr. Roger G. Stone
Department of Computer Studies
Loughborough University of Technology
LOUGHBOROUGH
Leicestershire
LE11 3TU
Tel: (0509) 222676
FAX: (0509) 610815
E-mail: R.G.Stone@uk.ac.lut

Newsletter Editor:
Dr. Jawed I. A. Siddiqi
Department of Computer Studies
School of Computing and Management
Sheffield City Polytechnic
100 Napier St
SHEFFIELD
S11 8HD
Tel: (0742) 720911 (direct: 533155/ msg: 533153)
E-mail: JS@uk.ac.scp.cms

Publicity:
Dr. Brian Q. Monahan
Department of Computer Science
The University of Manchester
Oxford Road
MANCHESTER
M13 9PL
Tel: (061) 275 6137
FAX: (061) 275 6280
E-mail: bqm@uk.ac.man.cs

Specialist Groups Management Committee Representative:
David Blyth
INCORD Ltd
15 Sherwood Avenue
Ferndown
WIMBORNE
Dorset
BH22 8JS
Tel: (0202) 896834
FAX: (0202) 894834

Dr. A. Jeremy J. Dick
Bull Research
68, Route de Versailles
78430 Louveciennes
FRANCE
Tel: +33 (1) 39 02 51 11
FAX: +33 (1) 39 02 49 77
E-mail: Jeremy.Dick@crg.bull.fr

Prof. Stephen J. Goldsack
Department of Computing
Imperial College
180, Queens Gate
Kensington
LONDON
SW7 2BZ
Tel: (071) 589 5111
FAX: (071) 581 8024
E-mail: sjg@uk.ac.ic.doc

Dr. Richard J. Mitchell
Department of Computing
Brighton Polytechnic
Moulsecoomb
BRIGHTON
Sussex
BN2 4GJ
Tel: (0273) 588641
FAX: (0273) 571701
E-mail: rjm4@uk.ac.bton.vms

Roger Shaw
Performance Technology Group
Lloyds Register of Shipping
Lloyds Register House
29, Wellesley Road
CROYDON
CR0 2AJ
Tel: (081) 681 4818
E-mail: ttercs@lloydreg.UUCP

Prof. Dan Simpson
Department of Computing
Brighton Polytechnic
Moulsecoomb
BRIGHTON
Sussex
BN2 4GJ
Tel: (0273) 600900 ext: 2273
FAX: (0273) 681752
E-mail: ds33@uk.ac.bton.vms

Dr. David Till
Department of Computer Science
City University
Northampton Square
LONDON
EC1V 0HB
Tel: (071) 253 4399
FAX: (071) 490 5028
E-mail: till@uk.ac.city.cs

Stephen Webster
Department of Computing and Cognition
Bournemouth Polytechnic
Fern Barrow
Dorset

FACS UPDATE

Fees

For 1992 these have been fixed at £25.00 and £10.00 (for members of BCS and sister societies).

Committee

This continues essentially as before but we have been joined by Roger Jones of ICL. Roger has given presentations at several of our workshops and it is a pleasure to have increased industrial involvement within the management of FACS.

Newsletter

We hope that this will now appear regularly and that this factor will encourage more submissions from members. Particular topics on which we are seeking articles are tools and education (case studies/experiences/integration of formal and informal subjects).

Membership

This has been stable for several years and we intend to launch a membership drive later in the year. A special issue of the Computer Bulletin will assist in this, and we are building up a list of institutional contracts through which we can publicise our events more widely.

The Journal

Subscribers to the journal will know that there are some developments in the pipeline, and these will take effect in Volume 4 (1992). Most importantly there will be an increase in frequency from 4 issues/year to 6 issues/year. This is an indication of the international success of the journal and is very welcome; what is perhaps less welcome is the accompanying increase in price. Volume 4 will be available to FACS members at £33.00 (full price £118.00). There will be a new section called "short communications" which will be overseen by Tim Denvir and we hope that members will be encouraged to submit articles for consideration. Particularly sought are short reports on the practical applications of formal methods, and 'letters' which pose technical questions.

John Cooke

B TUTORIAL

This was a three day event held at the Manchester University PEVE Unit on 15-17 April 1991. The first speaker was David Till who recapitulated our knowledge of formal systems. He covered the definition of a formal language, deductive apparatus, theorems, proofs and demonstrated these with a toy system containing premises, conclusions, derivations and so forth. He walked us through propositional symbols and rules of formation, truth tables, natural deduction and the tableau form of proof presentations.

After this introduction Jean-Raymond Abrial described the B proof system. He emphasised what B was not. B is not a proof tool or a proof checker, or most of the other things one might imagine it to be. He described B as "electronic paper designed to assist formal tree-structured proofs". He went through the low level syntax of the B language and explained applying a filter; this is like substituting a value for a variable but in textual form. In this way B can be used for forward and backward proofs and can be used to build automatic provers including proof by induction etc. Later when we saw demonstrations of the tool most people were impressed by its fast speed.

He then went on the describe the theory of abstract machines. An abstract machine has state and a specific number of operations which can be performed upon it. In some way it was reminiscent of VDM, I thought. He showed how the B language and tool could be used to build a tool-kit supporting the theory of abstract machines.

Next David Nielsen guided us through the tool-kit supporting abstract machines. This uses public domain software including, for example, Emacs, X-Windows, and Latex. The tool-kit uses Emacs to produce the abstract machine file, a type checker, and a normaliser to make a general substitution language version and an analyzer to produce a rule-base or collection of theories for proof obligations. These can be processed by the B tool or by the auto-prover acting on top of the B tool.

Another part of the tool-kit, the generator, produces a programming language called b0, bypassing the type checker and analyzer. The b0 text can be translated into another programming language such as C, Pascal, Ada or the Viper language.

A pretty printer is planned which will print out the text with annotations. This will output a Tex file.

The tool-kit can also handle refinements of abstract machines: inputting the specifications can automatically generate proof obligations which the automatic prover can prove. The tool-kit takes care of the details (provided adequate theories are supplied!).

Other lecturers stepped in to present various aspects of B and abstract machines: Peter Scharbach and Graeme Smith. Also interspersed with the lectures we had plenty of hands-on practice using Sparc stations in the PEVE Unit in the MU Computer Science Department. This was a particularly useful aspect of the tutorial. I found that after a little familiarisation with the supporting environment I was able to devise theorems and conjectures of my own and invoke the automatic prover to prove them. Having two people per work station was the right proportion I think. If people were at work stations singly they would have got lost that much more often, and if there had been more than two one would have been jostling continuously for a turn at having one's own hands on the equipment.

In all it was an intellectual demanding but very rewarding three days. A lot of documentation was supplied and plenty of hands on practice. The Computer Science Department supplied efficient tea and coffee facilities and wholesome lunches which were not so heavy, however, as to destroy one's concentration in the afternoons! Although the price of this tutorial was rather high for the usual FACS event, given the facilities and documentation supplied it seemed very reasonable value for money.

TIM DENVIR

FORMAL ASPECTS OF MEASUREMENT

This meeting was held on 3 May 1991 at the Polytechnic of the South Bank. Software metrics have for a long time been regarded as something of a black sheep by software formalists; the work has been perceived as empirical and lacking in scientific basis. By contrast the metricists have defended their position by saying "You make estimates about software projects that you are about to start and it is better to make these estimates on some evidence of the past than on none at all". More recently a more formal approach to software metrics has been adopted by certain researchers. It was to recognise this movement and to revisit the conflictual question of metrics versus formalism and indeed to examine whether there is any real conflict, that this meeting was organised. Virtually all the organisation was carried out on behalf of FACS by the Polytechnic of the South Bank, and we record our gratitude to them.

Abstracts were provided of all the talks and these are attached. The day started with Norman Fenton giving a talk in the nature of a survey. He explained that to establish a reasonable metric one had to identify and define the entity to be measured and the attributes that had to be measured, and then create a formal model for it. He emphasised that one had to be clear about whether the metric was going to be used for assessment or prediction. He talked about measuring aspects of the total development process rather than just, for example, code generation. At that point I did wonder how one would set about defining measurable attributes of the very early stages of requirements capture. Norman's book *Software Metrics a Rigorous Approach*, was on display and evoked a lot of interest.

The second talk was by Jim Bieman on measures of reuse in object oriented systems. He acknowledge the need to step towards a more formal, less ad hoc experimental approach. His talk envisaged a model comprising a library of components which were going to be used to build a system. He defined different degrees of reuse, depending upon how much the component was going to be modifies before reuse. These definitions were sufficiently broad to be able to include generic systems and Ada packages I think. He described how a partial kind of reuse, which he called "leveraged", could be facilitated by class inheritance in object oriented systems. His talk posed a lot of questions and was really a definition of what needed to be done in order that such metrication should be scientific. He did not, as far as I could tell, put forward theories of his own. Thus he was establishing the requirements rather than proclaiming a solution.

The second talk by Norbert Fuchs described part of the Cosmos project. He described flow graphs. These were very well defined but on the other hand seemed to be simply another metric of programme complexity. He flow graphs did not work particularly well for LOTOS specifications in their original form nor for recursion. So he modified them by assigning values to different kinds of nodes and demonstrated that he could then model these things rather more successfully. I felt that while he had devised a mathematically self-consistent model for a kind of metric, he had not appeared to address questions at a higher level concerned with what properties these metrics would usefully describe and how to determine whether or not the metric was a good description.

The third talk was by Austin Melton. I felt that his was by far the best analysis and used ideas of abstract models, theories and composition of mappings. He advocated that the theory should "tell you why" - in other words, I think, that it should be underpinned by a rational hypothesis. He also suggested that our measurement theories should be able to produce results which belonged to answer types which were not necessarily numeric. I felt very sympathetic with this point.

The fourth talk was by Horst Zuse. He claimed that metrics was both a mathematical and empirical science. His position was that any metrication scheme comprised a homomorphism from the system being measured to a scale. He defined different types of scale: ordinal, interval and ratio. These had various different kinds of properties such as being transitive, additive with respect to combinators, associative etc. He then classified various familiar metrics into these different categories of scale.

The next two talks were by Martin Shepperd and Dave Gustafson. At this stage my note taking began to flag so please see the abstracts for more details.

The last talk by Barbara Kitchenam advocated a more thorough statistical approach. Her talk was an emphatic critique of the traditional metrics work to date from the point of view of a thorough going statistician. This was delivered with a strength of conviction that left me, at any rate, ready to believe that the work to date had been from a statistical point of view entirely lacking!

We finished with a panel discussion. I shall not attempt to summarise this, if only because I was taking part in it! In all, I felt that this was a very successful day and a novel one. I certainly welcome the attempt to find a more rigorous and mathematical model of the subject of software metrics.

South Bank Polytechnic provided good organisation and a very good lunch for the delegates.

TIM DENVIR

British Computer Society
Formal Aspects of Computing Science Special Interest Group

One-day workshop on

Formal Aspects of Measurement

Friday 3rd May 1991, South Bank Polytechnic

Programme:

0930-1000: Registration

Morning session (Robin Whitty, chair)

1000-1005: Introductory remarks

1005-1030: Norman Fenton, City University
Software metrics: why a formal approach?

1030-1100: Jim Bieman, Colorado State University
Deriving measures of software reuse in object-oriented systems

1100-1115: Coffee

1115-1145: Norbert Fuchs, Alcatel Austria-ELIN
Language independent definition of axiomatic metrics

1145-1215: Austin Melton, Kansas State University
Designing and defining software metrics

1215-1245: Horst Zuse, Technical University of Berlin
Measurement theory applied to software metrics

1245-1400: Lunch

Afternoon session (Horst Zuse, chair)

1400-1430: Martin Shepperd, Bournemouth Polytechnic
Algebraic models, OBJ and metrics validation

1430-1500: Dave Gustafson, Kansas State University
Properties of software measures

1500-1530: Tea

1530-1600: Barbara Kitchenham, National Computing Centre
Never mind the metrics what about the numbers!

1600-1630: Panel discussion (Jim Bieman, Tim Denvir, Norman Fenton, Norbert Fuchs, Barbara Kitchenham)
Software measurement: a mathematical or an empirical science?

1630-1645: Closing remarks

Norman Fenton, City University
Software metrics: why a formal approach?

The observation of some very simple, but fundamental principles of measurement can have an extremely beneficial effect in the field of software measurement. Simply interpreting the formal definition of measurement in the software context leads to

1. Rationalizing and relating the various diverse software metrics activities

2. Practical help in constructing and validating software measures

3. The exposure of inconsistencies of some existing approaches in software measurement.

Any measurement involves an obligation to identify the entities of interest and the attributes of these to be measured. In software the entities may be classified as products, processes, and resources, while the attributes may be classified as internal or external to the entities. Next comes an obligation to to determine whether measurement is being used for assessment or prediction.

We shall look at some well-known approaches to software measurement within this framework, exposing both the good points and bad points. We shall briefly describe the relevance of measurement theory to software measurement.

Jim Bieman, Colorado State University
Deriving measures of software reuse in object-oriented systems

The analysis and measurement of current levels of software reuse is necessary to monitor improvements. This paper provides a framework for the derivation of measures of software reuse and introduces several definitions and attributes of potentially measurable reuse properties. The framework is applied to the problem of measuring reuse in object-oriented systems which support "leveraged" reuse through inheritance. I describe the importance of the perspective of the observer when analyzing and measuring reuse. Three perspectives are examined: the server perspective, the client perspective, and the system perspective. Each perspective gives the observer a different view of reuse in a system. These perspectives are especially relevant when applied to the problem of analyzing reuse in object oriented software.

Norbert Fuchs and Sieglinde Stainer, Alcatel Austria-ELIN

Language independent definition of axiomatic metrics

The aim of this paper is to present a way of having a more objective measurement of different attributes concerning both implementations as well as specifications of software programs. In order to gain this goal we are using a general basis on which all the information is stored, which is necessary and useful, when evaluating the corresponding implementation or specification by applying different metrics. The starting point, when introducing this general basis, was the usage of flowgraphs. As we all know, flowgraphs are a well-known and frequently used means for presenting the control flow of a program.

The theory developed by Fenton and Whitty is based on flowgraphs and can be seen as a means, with which one is able to generate a general basis of information presentation onto which we can apply metrics. As mentioned before, using a general information presentation irrespective of the corresponding language should provide the possibility of having a more objective measurement. Moreover, the definition of the different metrics is much easier, when using the theory of Fenton and Whitty. With the help of this theory, the way of presenting the information, which is measured afterwards, is done in a standardized way and the metrics definitions can always be done in the same structured manner.

But when using the flowgraphs theory of Fenton and Whitty in practice some disadvantages and even inaccuracies appear. Two main aspects will be discussed in this paper. The first disadvantage concerns the unique mapping of the control flow onto flowgraphs. In practice one can find a lot of examples where it is impossible to map the real control flow onto flowgraphs without losing valuable information. Some examples will be given in the paper. Second, as this theory is based on the structuredness defined by Dijkstra problems occurs whenever mapping a language onto the general basis, which is - seen from the point of view of Dijkstra's structuredness - not structured. But the attribute of being non structured in this way occurs in nearly every language. Whenever we try to map e.g. recursion onto flowgraphs we produce a non structured behaviour. Decomposing this flowgraphs results in the generation of decomposition trees with a lot of new big primes. This doesn't seem to be a problem, when seen from the theoretical point of view, but whenever this situation occurs in practice big problems occur, when doing the evaluation of the new big primes. It is quite simple to define metric values for the basic primes defined by Dijkstra but it happens frequently, that evaluating big new primes is not always possible in the accurate and adequate way we want to do it.

So, as a result we tried to modify the flowgraph theory of Fenton and Whitty in a way which prevents producing situations we can't handle in an adequate and correct way. We therefore introduced so-called 'descriptors'. A descriptor is a data structure which contains all the information necessary and useful for the metrics evaluation. Within this descriptor we store all the information concerning the control flow or information strongly related to the control flow of the corresponding program. As a consequence, the intermediate step of producing flowgraphs and decomposing them afterwards is no longer necessary. The basis on which we can apply all the different metrics is no longer the decomposition tree used in the Fenton and Whitty theory but a so-called 'descriptor tree'. This descriptor tree is quite similar to the former decomposition tree but doesn't contain any primes, only descriptors. Nevertheless, we still use the simple way of defining the metrics as proposed by Fenton and Whitty.

Within this paper not only the advantages of using descriptors are mentioned but also the syntax of the descriptor definition. As we want to use the same structure for different languages we have to define the syntax in a general way, which makes it applicable for different languages irrespective of whether they are implementation or specification languages. In order to represent the information contained in the descriptors in a properly structured manner we need to introduce a distinct classification, dividing similar language constructs into classes and subclasses. The classes are coarsely divided by semantic meaning, whilst the subclasses are a refinement corresponding to specific languages. Whenever the gain in important information is large enough, the descriptor is extended by an integer incorporating additional details useful for complexity measures. The classes defined so far are:

- Control Transfer: This class covers all types of control transfer including GOTO, IF, WHILE, procedure calls, and recursion.

- Exception Handling: This class covers any kind of exception handling.

- Parallel Processes: This class covers all the different aspects of concurrency, parallel process execution, and inter-process communication.

- Database Operations: This class covers different aspects of database operations.

Detailed information about this classification, the defined classes and subclasses as well as the additional integer arguments, is given in the paper.

At the end of this paper the two methods are compared with the result that the disadvantages and inaccuracies existent within the Fenton-Whitty theory, when used in practice, don't exist any longer, when using the modified method.

Austin Melton, Kansas State University
Designing and Defining Software Metrics

The purpose of this paper is to highlight the issues which need to be addressed when defining different types of software metrics or measures. By "types of software metrics" we mean metrics whose purpose is a) to measure features of a program document itself, b) to measure the use of a program document, and c) to estimate another metric. The basic issues when defining a metric of type (a) are clearly specifying the documents and features, or characteristics, to be measured; defining an appropriate abstraction of the documents; and deciding on the value or answer space. The additional basic issues when defining a metric of type (b) are deciding which document features play a vital role in the uses in question; and designing a theory which establishes what the role is. The additional basic issue when defining a metric of type (c) is designing a theory which establishes why the one metric should be an estimator of the other. Also this paper will address using measurement theory in designing software metrics.

Horst Zuse, Technical University of Berlin
Measurement theory applied to software metrics

During the last decade many software metrics were proposed by scientists and practitioners. However, scientists know that the scale level of a measurement process is important for the calculation of correct statistics, means and percentages. The problem of the scale level in the research area of software metrics is also discussed by other authors, like Mayrhauser, Harrison, Conte et al., etc. In this presentation the conditions for the use of software metrics as an ordinal, interval, ratio and absolute scale are shown. This concept is applied to the metric of McCabe. The conditions are given for the use of the metric of McCabe as an ordinal, interval, ratio and absolute scale. The consequences for other types of software metrics (design metrics, modularity metrics, etc.) are discussed.

Martin Shepperd, Bournemouth Polytechnic

*Algebraic models, OBJ and metrics validation**

A major problem in the field of software metrics is that much of the work might be characterised as speculative: that is, it requires considerably less effort to propose a metric than it does to produce a convincing validation of its utility. The outcome is a plethora of what may be regarded as putative metrics and a corresponding scarcity of properly validated metrics. This paper outlines a method for the formal evaluation of a software metric and its underlying model. This is based upon the specification of the model as an algebra and its desired behaviour as an associated axiom set. If these axioms can be proved to be invariant across the model, then the model may be considered to be valid with respect to its axioms. Where an axiom cannot be shown to be invariant this implies that either the model is anomalous or that the axiom was inappropriate. This approach is demonstrated with respect to a design metric based upon inter-module coupling. It is argued that this method of metric validation is a general one, which is capable of increasing confidence in the correctness of a metric particularly during the early stages of its development when empirical data may either be sparse or unavailable. A further benefit of this approach is that there exists a trivial transformation process from the algebra into OBJ, thus enabling the model to be animated. It is intended as a practical means whereby metrics workers can eliminate pathological models prior to embarking upon costly and time-consuming empirical validation exercises. Finally, it must be emphasised that this method should not supplant empirically based means of validation, rather that it should be employed as a complementary technique.

Keywords: Software metrics, measurement, validation, algebraic specification, software design.

---

*This work has been supported by British Telecom Research Labs., Martlesham Heath, Ipswich, IP5 7RE, England.

Dave Gustafson, Kansas State University
Properties of software measures

Software measures are important in managing software development. Selecting which measures to use is difficult. This paper describes desirable properties of software measures that have been mentioned in the literature. Attributes of the properties are discussed. The subsumes relationship between properties are investigated. A partial order of the properties is presented. Knowledge of these properties should help in selecting software measures.

Barbara Kitchenham, National Computing Centre
Never mind the metrics what about the numbers!

In this lecture, I argue that the practical use of software measurements rests on the discipline of statistics. In order to use software measurements we not only need the ability to measure product and process attributes, we also need the ability to describe the relationships and dependencies among attributes. Statistical methods allow us to formulate our models appropriately by making error terms explicit. Even more importantly, statistical techniques provide the mechanism to test the validity of our models objectively.

I will examine the mistakes that the software engineering community makes when it ignores the proper formulation and testing of non-deterministic models using examples from the domain of software cost estimation.

The empirical work described in this lecture is part of the ESPRIT MERMAID project. MERMAID is a joint collaborative project aimed at developing and automating improved methods of cost estimation.

THE UNIVERSITY OF SORRY

Academia ad Nauseam

Dear Dr. Siddiqi,

Congratulations! You have been selected out of a vast number of academics to receive a spectacular special offer! For a paltry £379[1] you may become the proud possessor of one of an *extremely limited edition* of

The Collected Internal Memoranda of F. X. Reid.

Yes! You, Dr. Siddiqi, may acquire for a paltry £435.85[2] a book that reveals the innermost preoccupations of one of the most singular minds of this century. Only in

The Collected Internal Memoranda of F. X. Reid

can you read - complete and unabridged the *maestro*'s magnificent sequence of requests for details of students absconding from first year FORTRAN tutorials, writing that recalls in its *crescendo* of pathos and invective the Philippics of Demosthenes. Only in

The Collected Internal Memoranda of F. X. Reid

can you find Reid's almost Wittgensteinian analysis of proposals for the unitisation of the Part Two B Artificial Intellegence Option. YES!

The Collected Internal Memoranda of F. X. Reid

is literally a treasure trove of administrative wisdom.

❊ WIT - POLEMICS - POSTMODERNISM - RATIONALISATION OF PHOTOCOPIER USAGE ○

---

[1] Or whatever the poll tax for Guldford will be in 1991...

[2] ...plus VAT...

and MUCH MUCH more, all in a book that no aspiring academic such as yourself, Dr. Siddiqi, should be without. Send your cheque, or cash (in used notes) for your *unique*, *extremely* limited edition of

The Collected Internal Memoranda of F. X. Reid.

SEE WHAT OUR READERS THINK!

"Engineering Applications Subject-area Subcommittee Meetings will never be the same again."

E. W. Dijskrasch

"Seldom has a Board of Studies been as brilliantly minuted."

F. X. Horra

"The interchange concerning the Dean's bi-annual research report questionnaire had me on the edge of my seat."

L. Lampost

"Lasciate ogni speranza, voi ch'entrate."

D. Alighieri

AND REMEMBER! If you send off NOW for your personalised edition of

The Collected Internal Memoranda of F. X. Reid,

then you, Dr. Siddiqi, will receive, ABSOLUTELY FREE, a copy of

TEENAGE MUTANT HERO TURTLE GRAPHICS.

Follow the exciting adventures of Carravagio, Alfresco, Cappucino and Arrivaderci as they battle their way through a defective graphics package to produce flyposters for the IT Soc annual pie 'n peas extravaganza!

This is an offer you can't refuse. Send off NOW to

Dirk Thrust,
(The F. X. Reid Offer),
Unisor Enterprises plc,
3, Sherborne Court,
The Mount,
Guildford.

ICL Z PROOF TOOL NEWSLETTER NO. 1 - 16 May 1991

1. CONTENTS

2. INTRODUCTION

This is the first of a series of newsletters which we will produce to keep prospective users and other interested parties in touch with the status of our work on Z proof support. We are eager to receive feedback from any interested parties on what we are doing and on how this relates to their needs or ambitions in this area. To this end we intend to adopt as open a policy as is consistent with retention of IPR in the developments. When drafts of specifications or user documentation become available these will be announced in this newsletter, and we will make these available by email or hard-copy. Later sections describe how we propose to distribute the documents, and what documents are currently available.

The development of the tool is being undertaken partly under a DTI supported collaborative research project known as the FST project and entitled "Foundations and Tools for Formal Verification". ICL's partners in this are Program Validation Limited, and the Universities of Cambridge and Kent.

This newsletter contains a description of the kind of tool we are developing, a status report on the development, and some information about draft formal specifications of critical aspects of the proof tool.

The fact that any development is described in this newsletter does not constitute an undertaking by ICL that the development will necessarily be continued to completion.

3. DESCRIPTION OF TOOL

The Z proof tool will support syntax checking, type checking and formal proof in Z. This support is provided in a multilingual proof support environment based on HOL [3] and implemented in standard ML [2] following the LCF paradigm [1].

The following sections provide a description of the technology underlying the Z proof system, its use in support of Z and some other useful characteristics of the resulting system.

3.1. Technology Base

The ICL proof system is based on well proven languages and techniques. These are being re-engineered by ICL and applied to new problems.

The main objectives of this re-engineering activity are as follows:

* product quality implementation
* high assurance of soundness of proof system
* improved proof development productivity
* proof support for Z

To ensure good quality in the ICL HOL system, ICL has first implemented a "prototype", which has now been in use on the project for about a year. This was not a "port" of the Cambridge HOL system to standard ML, it was a prototype for a future ICL implementation not intended to precisely replicate the functionality of the Cambridge HOL system. The first version of product quality ICL HOL is now designed and implemented and is expected to be completed in Q3 1991.

High assurance of soundness is to be achieved by:

(a) Faithful adherence to the "LCF paradigm" [1](using protected types for terms and theorems, and not allowing any loop holes).

(b) Minimisation of code critical to soundness of proof system (e.g., the parser, which is critical in Cambridge HOL, is not critical in our system, it is implemented in ML and uses the standard term constructors).

(c) Formal treatment of the critical parts of the system. Formal specifications released for public scrutiny (just in case someone else is willing to read them). Partial verification of the design of critical subsystems.

Early versions of the system will benefit primarily from greater regularity in the proof support facilities. Later versions will contain the results of more aggressive attempts to implement state-of-the-art proof automation. The objective is to get much closer to being able to prove "obvious" lemmas fully automatically. Later versions will also provide effective use of up-to-date HCI facilities (windows mice and menus).

We are now expecting to achieve better proof support for Z than we had thought possible when the project was initially proposed. This will be achieved by treating Z as a "secondary" object language, using a semantic embedding of Z into HOL. The system will support multiple object languages each with their own parsers, type checkers, pretty printers and derived inference rules. Support for Z will cover the full language as presently defined using a semantics which is consistent with the available documentation [5,6] for semantically well formed specifications.

3.1.1. LCF paradigm

The paradigm adopted is the LCF paradigm, due to Robin Milner and others who developed the LCF system at Edinburgh University [1].

The main distinctive feature of the LCF paradigm is that it supports proof development by programming in a typed functional programming language. This functional language is referred to as the "metalanguage" to distinguish it from the language of the logic in which the proofs are conducted, which is known as the object language. The LCF paradigm permits the metalanguage to be made available to the user, for programming proof computations, without risk of the system accepting unsound derivations.

3.1.2. Metalanguage - Standard ML

The metalanguage, which is both the implementation language of the system and a language available to the user for interacting with the system, is standard ML [2], a modern functional programming language. Standard ML is a practical compromise between pure functional languages, which bar all side effects, and imperative languages which depend too heavily upon side effects. The effect is that most of the system is implemented in a pure functional code, but where this proves unduly awkward or delivers inadequate performance imperative features can be employed.

3.1.3. Primary Object Language - HOL (Higher Order Logic)

The primary object language of the system is HOL [3]. HOL is a polymorphic variant of Church's simple type theory [7]. It was originally implemented as an LCF-style proof tool by Mike Gordon and others at the University of Cambridge in 1985 using a variant of ML produced jointly by Cambridge and INRIA. Since then it has been re-implemented in standard ML by Konrad Slind of Calgary University [8] and (independently) by ICL at Winnersh. The Calgary implementation follows closely the Cambridge system while providing a number of additional facilities (e.g. Knuth-Bendix completion and AC unification). The ICL implementation, while supporting the same logic, follows the Cambridge implementation less closely. It is intended to be used for support of more complex languages such as Z.

An important merit of HOL is its combination of power and simplicity. It is logically similar in strength to Zermelo set theory, (or the system of Russell's "Principia Mathematica"), it has a very simple decidable polymorphic type system, and the abstract structure of the underlying types and terms involves just six constructors. This considerably simplifies the task of writing general purpose proof algorithms in the metalanguage.

3.1.4. Secondary Object Languages

Because of its simplicity and power HOL is a good logic for supporting reasoning about scripts written in other languages, which we will call secondary object languages. The underlying term structure which is used for formal reasoning in HOL is suitable for representation of secondary languages with more elaborate concrete syntax. This representation can often be done in a semantically faithful way, so that the semantic properties expected of the relevant constructs in the secondary language are possessed by the underlying HOL representation. If a representation having this property is adopted, reasoning in the secondary object language can be undertaken using derived rules in HOL.

The tool is intended for use as a multilingual proof support environment in which secondary object languages are supported by such semantic mappings into the primary object language, HOL. Machinery used in the construction of parsers, type checkers and pretty printers for the supported languages will be available for reuse by anyone wishing to provide support in this framework for further languages.

Similar approaches to support of other languages have been adopted by Mike Gordon using Cambridge HOL (see e.g. [4]). The first application of the ICL HOL system to support of other languages is to the specification language Z.

3.1.5. Interface Style

The prototype ICL HOL system supports the same style of interface as the Cambridge HOL system. This is an interactive teletype interface, normally run under some window management system. The ICL prototype operates with an extended character set including the special characters needed for Z.

At version 1 this interface will not be substantially different from the prototype. The main changes (relative to Cambridge HOL) will be:

(1) A more regular, transparent and comprehensive basic set of inference facilities so that the user needs to remember less, finds it easier to look up what he can't remember, and less frequently finds that what he wants is missing.

(2) Uniform general purpose support facilities for secondary languages.

The underlying standard ML implementation will be the PolyML standard ML compiler, originally developed by Dave Matthews at Cambridge University and now marketed by Abstract Hardware Limited. This provides full support for X Windows, though exploitation of this by ICL HOL version 1 will be limited. Later versions of ICL HOL and the Z proof tool will benefit from a more thorough review of interfacing issues.

3.2. Support for Z

ICL support for the Z specification language will be provided in a multilingual interactive environment providing proof support via the LCF paradigm. This environment will provide standard ML (which may be used for programming derived proof rules or other user extensions to the system, or may be used for rapid prototyping to clarify requirements), HOL (Higher Order Logic) which may be used for specification and/or formal proof, and Z.

Support for Z will include interactive or batch syntax and type checking of scripts which are LaTeX documents using a character set extended by the special characters required in Z. The tool will support formal reasoning in Z using proof rules and tactics which are derived in HOL from an encoding of the semantics of Z in HOL. The details of these mappings and derivations are not visible to the naive user, but are visible to advanced users who may wish to extend the automatic proof facilities by programming sophisticated proof algorithms in ML (though knowledge of the mappings is not essential for tactical programming). The semantic mappings are all accomplished using only conservative extensions of HOL, and therefore the proof system for Z will be consistent if that for HOL is.

Both forward proof in Z and goal-directed proof will be supported in a manner similar to that for HOL. The HOL rewriting facilities will be extended to cope with conditional rewrites using equations in Z, which are often quantified over sets rather than types. Z specifications will be assumed to be intended to be conservative on a paragraph-by-paragraph basis and proof obligations will be enforced by storing consistency caveats in the theory. Support will be provided for automatic proof of such consistency caveats, and this will also result (when successful) in proofs of "semantic well formedness" (lack of undefined expressions or subexpressions) which will facilitate subsequent reasoning. Extraction and simplification of pre-conditions will also be supported.

Fuller details of the Z proof style await further progress in the current prototyping activity.

3.3. Other Features

3.3.1. Prototyping

The use of standard ML as an interactive metalanguage, with X Window support in ML suggests that this environment may be suitable for fast prototyping of certain kinds of applications. In this single environment it will therefore be possible to use prototyping to clarify requirements and to explore HCI issues, as well as using formal techniques to precisely define critical requirements and for reasoning about the ability of particular designs to meet those requirements. This combination of fast prototyping, formal modelling and formal proof will offer flexibility to those wishing to apply formal techniques selectively and to best effect.

3.3.2. Educational Aspects

By offering interactive computational support for a modern functional programming language (ML) a powerful classical logic (HOL), and a rich specification language (Z), this tool (already) has considerable potential in support of training in functional programming, discrete mathematics and formal methods. This could be relevant in undergraduate teaching, industrial training or in self teaching packages. Standard ML with X Windows will provide an excellent and productive implementation medium for relevant course material. ICL is currently considering the best way to exploit and extend this potential, and wishes to encourage others to do likewise.

3.3.3. Artificial Intelligence

This system represents a "logic/programming" environment in which the strength and expressiveness of the logic has not been constrained by a requirement for all expressible functions to be computable. A separation is retained between domain knowledge and search strategies, the former being expressed in higher order logic, the latter in a computationally more efficient higher order functional programming language (ML).

The AI component of our work on this tool will be confined to the incorporation of modern proof search techniques drawn from the literature. There are few limits on how many approaches to automatic proof can be supported, since ICL or other users of the system can program new methods in standard ML, where appropriate reusing facilities already implemented in previous proof algorithms.

The question whether this system would provide a good basis for implementation of logic databases, expert system shells or any application oriented AI work is one which we will not ourselves be able to investigate.

4. STATUS OF TOOL DEVELOPMENT

4.1. ICL HOL prototype

A prototype of the ICL HOL system has been in use on the project since May 1990, and will be replaced by production quality system during 1991. This system has been used for research in proof automation, for prototyping Z proof support, and for generally clarifying our ideas on what the ICL HOL system should look like.

The libraries and superstructure are less extensive than those in the latest Cambridge HOL systems, but the integrity of the implementation is good. This version of HOL will not be made generally available.

4.2. ICL HOL "product quality"

A product quality version of ICL HOL is now in production and is expected in suitable condition for limited external release in the third quarter of 1991.

4.3. ICL Z proof support prototype

ICL now has a prototype Z proof system based on ICL HOL.

The prototype Z support system currently provides syntax and type checking (interactive and batch) and permits proof via the semantic mappings. It covers the full Z language as specified in "The Z notation" (with one or two minor omissions of "sugar"), and the complete library as specified in that book is available as a theory (at present containing only the definitions). Implementation of derived rules for reasoning in Z without resort to the mappings began in April 1991.

This prototype is for research purposes and will not be made generally available.

4.4. ICL Z proof support (product quality)

ICL intends to produce a Z proof support tool suitable for use on projects within ICL and for external release as soon as possible. This will definitely not be before mid 1991, and may not be until 1992. We intend to support "standard Z" if a BSI or ISO standard emerges provided that semantic issues are clarified in good time. However, if the a standard is slow in emerging or imprecise on semantics we will not be inhibited from producing the tool. We do not need to wait for academic consensus on proof rules (except insofar as this affects consensus on semantics), since the Z proof rules are derived in HOL using a semantic mapping (this will not be visible to the average user).

5. LICENSING ARRANGEMENTS

We hope to make the tool available for academic research at as low a cost as possible. Early versions of the system will be available only with the AHL PolyML standard ML compiler and prices to academic sites not already licensed for PolyML will therefore depend upon negotiations currently in progress with AHL. Fees for licenses for commercial use will be determined at a later date.

6. CURRENTLY AVAILABLE SPECIFICATIONS

The documentation currently available in a form suitable for external distribution consists of formal specifications of critical aspects of the HOL system (see 7.1) and a theoretical paper on Z free type definitions (see 7.2).

The formal specifications of HOL are literate scripts in which the formal material is in HOL using a concrete syntax with Z-like non-ascii characters and in a style which is similar to a Z specification written in a functional style using axiomatic definitions. These should therefore be mostly intelligible both to people who are familiar with HOL, and to those familiar only with Z.

6.1. Formal Specification of HOL

In order to base the ICL HOL system on a theoretically sound basis which is as rigorous as possible, we have produced a formal specification in HOL of the language, its semantics, and its deductive system. This theoretical material has then been used to formulate the critical requirements on an abstract model of the proof development system.

These specifications are available to anyone who is interested. They comprise five documents identified by our internal references as follows:

SPC001 (24 pages)	HOL formalised: Language and Overview
SPC002 (24 pages)	HOL formalised: Semantics
SPC003 (48 pages)	HOL formalised: Deductive System
SPC004 (14 pages)	HOL formalised: Proof Development System
SML027 (17 pages)	HOL formalised: Support theory for SPC001

These are "literate scripts" which are processable by the prototype ICL HOL system and cause a hierarchy of theories to be established with the following structure:

```
        (SML027)
            |
            |
         SPC001
          /   \
         /     \
     SPC002   SPC003
         \     /
          \   /
         SPC004
```

In general, except possibly for SML027 (which contains definitions of certain basic set theoretic apparatus which you can probably guess the meaning of) if you don't have them, it is desirable to have to hand all the documents on which a document depends if you want to be able to read and comprehend it.

Notes on these documents follow:

6.1.1. SPC001 - Language and Overview

This gives and overview of the document suite and contains the formal specification of the language and of other syntactic structures, such as theories, which are needed elsewhere in the specification.

6.1.2. SPC002 - Semantics

This contains the specification of the semantics. The semantics is essentially a generalisation of the standard set theoretic semantics for simple type theory to cover the polymorphism found in HOL. The treatment is intended to give the "same" semantics as the treatment in ordinary set theory due to A. Pitts which can be found in the Cambridge HOL documentation. However the use of HOL rather than ZF as the metalanguage makes a significant difference at a detailed level.

6.1.3. SPC003 - Deductive System

This contains the specification of the deductive system. The treatment of the inference rules includes a full account of substitution and type instantiation.

6.1.4. SPC004 - Proof Development System

This contains the specification of an abstract model of the proof development system and of its critical properties. Roughly speaking a proof development system is taken to be an automaton whose state can be interpreted as a hierarchy of HOL theories. Three critical properties are identified. The first two are the semantic and syntactic characterisation of the assertion that the system cannot infer theorems which do not follow from given axioms. The third assertion reflects a strongly felt preference for conservative extensions; it effectively demands that any extension which the system claims is conservative really is conservative.

6.1.5. SML027 - Support theory for SPC001

This contains definitions of some library material which was not available on the prototype when the specification was developed.

6.2. WRK016 - On Free Type Definitions in Z

Recent correspondence in the Z forum has considered the issue of the conservativeness of the free type construct in Z. The original question asked whether free type definitions which met the criterion for consistency given in "The Z notation", [] were conservative over Zermelo set theory (i.e. Zermelo without the axiom of replacement). The main purpose of this paper is to show that the answer to this question is "yes" given the axiom of choice).

The question is of relevance to attempts to give specification and proof support for Z in system such as Mike Gordon's HOL, since to support Z free type definitions which were not conservative over Zermelo would require non-conservative extensions to HOL.

7. DISTRIBUTION OF DOCUMENTS

If you want to receive any of the above documents you should send a request to me, Roger B. Jones stating:

(1) the reference numbers of the documents

(2) your email address

(3) your full postal address

(4) whether a LaTeX "dvi" file is acceptable to you

I will then send you the requested documents either electronically or by ordinary mail.

Requests should be sent to me at any of the following addresses (in order of preference, use the first one that your mailer understands):

(1) R.B.Jones@win0103.uucp

(2) R.B.Jones@win0103.icl.icl.gold-400.gb

(3) rbj@win.icl.co.uk

(4) Mr. R.B.Jones
International Computers Limited
Eskdale Road
Winnersh
Wokingham
Berks RG11 5TT
ENGLAND, UK

Please let us know (preferably electronically) if you find any problems with the documents, or have any comments.

The preferred method of distribution is by electronic transmission of LaTeX "dvi" files. If anyone who would like to receive a document is not capable of receiving it in this way we will send hard copy by ordinary mail.

Documents will be transmitted as LaTeX "dvi" files compressed (using "compress") and encoded (using "uuencode"). Where necessary they will also be split to permit transmission.

To print these documents it will therefore be necessary to have the LaTeX software and appropriate fonts.

"appropriate fonts" in this case means the standard "cm" + "l" (computer modern) fonts and also the "msx" and "msy" fonts. All of these are on the normal LaTeX distribution tapes.

To recover the "dvi" file it is necessary first to use the UNIX utility "uudecode" and then "uncompress".

We have not previously distributed documents by this means and so there may be teething problems. You may therefore experience some delay before receiving the documents you request. If we have any serious difficulty we will resort to hard copy distribution.

8. <u>DISTRIBUTION OF NEWSLETTERS</u>

If you are on my distribution list and don't want to be, or if you are not on it and want to be, please mail me at the above address.

9. REFERENCES

[1] M. Gordon, R. Milner, and C.P. Wadsworth, "Edinburgh LCF a Mechanised Logic of Computation", lecture notes in computer science, Springer-Verlag, 1979

[2] Robin Milner, Mads Tofte, and Robert Harper, "The definition of Standard ML", MIT press, 1990.

[3] M.J.C.Gordon, "HOL: A Proof Generating System for Higher Order Logic", in G.Birtwhistle and P.A.Subrahmanyam, editors, VLSI Specification, Verification and Synthesis, Kluwer, 1987.

[4] M.J.C.Gordon, "Mechanising Programming Logics in Higher Order Logic", in G.Birtwhistle and P.A.Subrahmanyam, editors, proceedings of the 1988 Banff Conference on Hardware Verification, Springer-Verlag, 1988.

[5] M.J.Spivey, "Understanding Z", Cambridge University Press, 1988.

[6] M.J.Spivey, "The Z notation - a reference manual", Prentice-Hall 1989.

[7] A. Church, "A Formulation of the Simple Theory of Types", Journal of Symbolic Logic 5, 1940.

[8] Konrad Slind, "An Implementation of Higher Order Logic", Research Report No. 91/419/03, Department of Computer Science, The University of Calgary.

jonesrb/docs/1048

BCS–FACS RAISE TUTORIAL

30th September – 2nd October 1991
Post Experience Vocational Education (PEVE) Unit
Department of Computer Science
The University, Manchester

1 Introduction

RAISE, standing for the Rigorous Approach to Industrial Software Engineering, comprises a formal specification language RSL, a development method and a comprehensive supporting tool set. RAISE was developed within the Esprit I project 315. This tutorial aims to provide a general appreciation of RSL and the RAISE development method and will also introduce participants to the RAISE toolset. Information from LaCoS, an Esprit II project aimed at applying the RAISE method to industrial scale problems, will also be presented.

2 Background Information

2.1 The RAISE Specification Language – RSL

RSL is a *wide spectrum language* in the sense that it is possible to specify all stages of the software development process using it. RSL contains facilities that support very abstract and general specifications as well as very implementation oriented specifications containing facilities similar to those found in procedural programming languages. RSL is probably the most generally applicable formal specification and design language available today. RSL encompasses and integrates the major styles for formal specification and design that have emerged over the last two decades:

- algebraic specification
- model-oriented specification
- modularisation and parameterisation at the structuring level
- axiomatic as well as explicit definitions
- applicative, imperative and concurrent styles

2.2 The RAISE Method

The RAISE method is based on the notion of stepwise refinement whose basis can be summarised as follows:

- Each step starts with a description of the software and produces a new one which is in some way more detailed (or concrete)

- The result of each step is not only more detailed but also in some way conforms to the previous one, so that it can be used to replace it
- Refinement typically involves both algorithm and data, since a change in one normally involves a change in the other

This basis is taken into account in RAISE developments where initial abstract specifications are successively refined by a process of commitment in which degrees of freedom are removed. In each step data structures and/or control structures are elaborated. Development steps also involve justifications that each new specification, or combination, in some sense is a correct development of the previous one.

2.3 RAISE Tools

RAISE has a collection of tools for manipulating a variety of entities that are relevant during a development process, for example, modules, and relations between modules. Individual tools for manipulating such entities are centered around the RAISE Library, which is a specialised database system. The RAISE tools include a Module Editor, specialised Entity Editors, a Library Query Editor, Justification Tools and Translators. Translators are available for generating Ada and C++ from low level designs expressed in RSL.

3 The Tutorial

The tutorial aims to provide a general understanding of the RSL specification language and how it and the RAISE method are combined, either rigorously or formally, in the software production process. The RAISE tools will be demonstrated showing how they may be used to support the development of software systems.

4 Practicals

The tutorial will be limited to 24 people. Tutorial sessions will be divided between lectures and practical sessions. 12 Sun workstations will be available and these will be shared one between two course members. The RAISE tools are commercially available and information will be provided during the tutorial concerning terms and licencing arrangements.

5 Lecturers

The course will be given by:

Chris George	CRI (Denmark)
Søren Prehn	CRI (Denmark)
Roger Shaw	Lloyd's Register

6 Cost

The cost of the tutorial will be £220.00 per person for non FACS members and £190.00 per person for FACS members (VAT is included in these prices). This charge includes copies of the course material together with coffee, lunch and tea on each of the three days of the tutorial. The cost of accommodation is **not** included in this charge. Accommodation may be secured at the Manchester Business School (subject to availability) or at a local hotel.

7 Booking Procedure

Please complete the attached form and return it by Friday 23rd August 1991.

BCS – FACS RAISE TUTORIAL

30th September – 2nd October 1991
PEVE Unit – Department of Computer Science
The University, Manchester

REGISTRATION: The registration fee, including VAT at 17.5%, is £220.00 for non FACS members and £190.00 for FACS members. *Administration costs make it necessary to surcharge these prices by £10.00 for applications not accompanied by a payment.*

NAME:

ADDRESS:

TELEPHONE NUMBER:

AMOUNT ENCLOSED:

Cheques should be made payable to **BCS FACS** and sent, by Friday 23rd August, together with this form to:

Mr Roger Shaw
Performance Technology
Lloyd's Register of Shipping
Lloyd's Register House
29, Wellesley Road,
Croydon
CR0 2AJ

Telephone 081 681 4818

Please use a separate form for each person registered. (Photocopies of this form are quite acceptable).

ACCOMMODATION: Accommodation (bed and breakfast) may be secured at the Manchester Business School (061 275 6333) for £40.25 per night. Those wishing to attend the tutorial and stay at the Manchester Business School should arrange their own bookings and are responsible for paying their own bills. Alternatively, other accommodation within the Manchester area may be secured.

FORTHCOMING EVENTS

1991

Date: **September 3 - 6**
Title: **Category Theory and Computer Science**
Location: Ecole Normale Supérieure, Paris, France.
Contact: David Pitt, Department of Mathematics, University of Surrey, Guildford, Surrey GU2 XH, UK.
Email: dhp@cs.surrey.ac.uk.
Local arrangements: Pierre-Louis Curien, LIENS, 45 rue d'Ulm, 75230 Paris Cedex 05, France.
Email: curien@dmi.ens.fr.

Date: **September 9 - 13**
Title: **16th International Symposium on Mathematical Foundations of Computer Science**
Acronym: MFCS'91
Location: Warsaw, Poland.
Sponsors: Institute of Computer Science of the Polish Academy of Sciences and the Institute of Informatics of Warsaw University.
Contact: P. Chrzastwoski-Wachtel and A. Tarlecki, MFCS'91, Institute of Computer Science, Polish Academy of Sciences, PKiN, P.O. Box 22, 00-901 Warsaw, Poland.

Date: **September 9 - 13**
Title: **Fundamentals of Computation Theory**
Location: Altenhof, near Berlin, Germany.
Acronym: FCT '91.
Sponsor: SIEMENS AG.
Contact: B. Molzan (secretary), B. Graw, U. Schäfer, FCT '91, Karl-Weierstraß - Institut für Mathematik, PF 1304, 0-1086 Berlin, Germany.

Date: **September 16 - 18**
Title: **Software Engineering for Real Time Systems**
Location: Royal Agricultural College, Cirencester, UK.
Sponsor: IEE.
Organisers: Institution of Electrical Engineers.
Contact: Conference Services, The Institution of Electrical Engineers, Savoy Place, London WC2R 0BL, UK. Tel. 071 240 1871 Ext. 222. Telex. 261176 IEE LDN G. Fax. 071 240 7735.

Date: **September 18 - 20**
Title: **Working Conf. on Security and Reliability in Distributed Systems**
Location: Prince Edward County, Ont., Canada.
Sponsor: IFIP.
Contact: Stewart Lee, Computer Systems Research Inst., D.L. Pratt Bldg., Univ. of Toronto, 6 King's College Rd., Toronto, Ont., Canada M5S 1A4, 'phone (416) 878-5035, fax (416) 978-4765.
Email: stew@hub.toronto.edu.

Date: **September 22 - 25**
Title: **6th Knowledge-Based Software Engineering Conference**
Acronym: KBSE-6.
Location: Syracuse, N.Y., USA.
Contact: Peter G. Selfridge, AT&T Bell Lab, Rm. 3C-441, Murray Hill, NJ 07974; 'phone: (201) 582-6801.
Email: pgs@research.att.com.

Date: **September 23 - 25**
Title: **First Int'l Workshop on the Economics of Design and Test**
Location: Austin, Texas, USA.
Sponsor: SIGDA.
Contact: Sarma Sastry, Dept. of EE Systems, SAL 340, Univ. of Southern Calif. at Los Angeles, Los Angeles, CA 90089-0781; 'phone (213) 743-0528.
Email: sastry@vishnu.usc.edu.

Date: **September 23 - 27**
Title: **Computational Intelligence '91**
Location: Milano, Italy.
Sponsor: PC and VG in coop. with SIGART.
Contact: Mr. Giorgio Valle, Univ. Degli Studi di Milano, via Moretto, 00 Brecia 9, Milano 20133, Diparto, Italy; 'phone 0039 2 2772278.
Email: valle@imiucca.

Date: **September 23 - 27**
Title: 3rd International Workshop on Foundations of Models and Languages for Data Objects
Location: Aigen, Austria.
Sponsor: GI-Arbeitskreis, Grundlagen von Informationsysetem, Gesellschaft fur Informatik (GI) in coop. with EATCS.
Contact: G. Saake, Informatik, Abt. Datenbanken, Techn. Univ., Postfach 3329, D-W3300 Braunschweig, Germany; 'phone: ++49 531 391 3267; fax: ++49 531 391 4577.
Email: saake@infbs.uucp or saake@dbsinf6.bitnet.

Date: **September 24 - 27**
Title: **Theoretical Aspects of Computer Software**
Location: Tohoku University, Sendai, Japan.
Sponsors: Tohoku University, Information Processing Society of Japan, IEEE, ACM SIGACT, Association for Symbolic Logic.
Contact: Prof. Takayasu Ito, Department of Information Engineering, Tohoku University, Sendai, Japan 980.
Email: ito@ito.ecei.tohoku.ac.jp.
Fax: 81 22 267 4404.

Date: **September 30 - October 2**
Title: **10th Symp. on Reliable Distributed Systems**
Location: Pisa, Italy.
Contact: Luca Simoncini, IEI-CNR, Via S. Maria 46, 56100 Pisa, Italy; 'phone: 39 (50) 553-159, fax: 39 (50) 554-342.
Or: Ozalp Babaoglu, Dip. di Matematica, Univ. di Bologna, Piazza di Porta S. Donato, 5, 40127 Bologna, Italy; 'phone: 39 (51) 354-430, fax: 39 (51) 354-490.
Email: ozalp@dm.unibo.it.

Date: **October 2 - 4**
Title: **Foundations of Computer Science**
Location: San Juan, Puerto Rico.
Sponsor: IEEE
Contact: Local Arrangements Chairs: Tom Leighton, Laboratory for Computer Science, M.I.T., Cambridge, MA 02139, USA.
Or: Alok Aggarwal, T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA.

Date: **October 6 - 9**
Title: **1st International Conference on Software Quality**
Acronym: ICSQ 1
Location: Dayton, Ohio, USA.
Contact: John Lowe, LCS SQA 4020 Executive Dr., Dayton, OH-45430; 'phone (513) 429-6458; fax: (513) 429-6368.
Paper Submission Details:
Submit four copies of 300-word abstract, panel proposal or tutorial proposal to John Lowe.

Date: **October 6 - 11**
Title: **OOPSLA 91**
Location: Phoenix Convention Senter, Phoenix, Ariz., USA.
Sponsor: SIGPLAN.
Contact: John Richards, IBM TJ Watson Research Ctr., PO BOX 704, Yorktown Heights, NY 10598; (914) 784-7731.
Email: jtr@ibm.com.

Date: **October 7 - 8**
Title: **Fifth SEI Conference on Software Engineering Education**
Acronym: CSEE 91
Location: Pittsburgh, USA.
Contact: James E. Tomayko, Software Engineering Inst., Carnegie Mellon Univ., 4500 Fifth Ave., Pittsburgh, PA 1521303890, phone (412) 268-6806, fax (412) 268-5758.
Email: jet@sei.cmu.edu.

Date: **October 7 - 11**
Title: Computer Science Logic
Location: Berne, Switzerland.
Acronym: CSL '91.
Contact: Prof. Dr. G. Jäger, CSL '91, Institut für Informatik und angewandte Mathematik, Universität Bern, Länggassstrasse 51, CH-3012, Berne, Switzerland.

Date: **October 8 - 10**
Title: **4th Symposium on Testing Analysis and Verification**
Acronym: TAV-4.
Location: Victoria, British Columbia, Canada.
Sponsor: SIGSOFT.
Contact: William E. Howden, CSE, UCSD, La Jolla, CA 92093; phone (619) 534-2723.
Email: howden@cs.ucsd.edu.

Date: **October 14 - 17**
Title: **Conference on Software Maintenance**
Location: Sorrento, Italy
Sponsors: IEEE Computer Society - Technical Committee on Software Engineering, The Institute of Electrical and Electronics Engineers, Inc.
Contact: John Munson, Division of Computer Science, University of West Florida, Pensacola, FL 32514, 'phone: (904) 474-2989.
Email: jmunson@uwf.bitnet.
Or: Roberto Ciampoli, Olivetti Information Services, Vie Croce 19, 00142 Rome, Italy, 'phone: +39 (6) 5411552, fax: +39 (6) 5415239.

Date: **October 14 - 18**
Title: **Provably Correct Systems Symposium**
Location: Gl. Avernæs, Denmark.
Acronym: ProCoS.
Contact: Mrs. Annie Rasmussen, Dept. of Comp. Sci., Bldg. 344, Techn. Univ. of Denmark, DK-2800 Lyngby, Denmark; Fax: +45 42 884530, 'phone: +45 45 933332, Telex: 37529 dthdia dk.
Email: procos@id.dth.dk.

Date: **October 16 - 19**
Title: **5th International Symposium on Methodologies**
Location: Charlotte, N.C., U.S.A.
Acronym: ISMIS-91.
Sponsor: Univ. of North Carolina.
Contact: Bill Chu, Dept. of Comp. Sci., Univ. of North Carolina, Charlotte, NC 28223; 'phone (704) 547-4568.
Email: billchu@unccvax.uncc.edu.

Date: **October 21 - 22**
Title: **First Int'l Conf. on the Software Process**
Location: California, USA.
Acronym: ICSP 1.
Contact: ICSP 1, PO BOX 3521, Boulder, CO 80303; 'phone: (303) 499-4782.
Email: icsp1@sda.com.

Date: **October 21 - 24**
Title: **Third European Software Engineering Conference**
Contact: Alfonso Fugetta, CEFRIEI, c/o AICA-ESEC '91, P.le Rodolfo Morandi 2, 1.2021 Milan, Italy.
Email: alfonso@imicefr.bitnet.
Acronym: ESEC 91
Location: Milano, Italy.
Sponsors: AFCET et al.

Date: October 21 - 25
Title: **VDM'91**
Location: Leeuwenhorst Congress Center, The Netherlands.
Contact: Hans Toetenel, OC-chair, Delft University of Technology, PO BOX 356, 2600 AJ Delft, The Netherlands.
Email: toet@dutiaa.tudelft.nl.
Or: PC-chair - Soren Prehn at sp@cpe.csd.cri.dk.

Date: October 25 - 26
Title: **Sixth Int'l Workshop on Software Specification and Design**
Location: Como, Italy.
Sponsor: SIGSOFT and IEEE-CS.
Contact: Jean-Pierre Finance, CRIN Univ. de Nancy 1, Campus Scientifique, BP 239, 54506 Vandoeuvre les,; 33 83 91 21.
Email: finance@loria.crin.fr.

Date: October 28 - 31
Title: **Int'l Logic Programming Symp.**
Location: San Diego, California, USA.
Acronym: ILPS 91.
Cosponsors: Assoc. of Logic Programming et al.
Contact: Vijay Saraswat, Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304; 'phone: (415) 494-4747, fax: (415) 494-4334.
Email: ilps91@parc.xerox.com.
Or Local Arrangements:
Steve Taylor, CalTech, USA.
Email; steve@visi.caltech.edu.

34

Date: November 19 - 22
Title: **Fourth International Conference on Formal Description Techniques**
Location: Sydney, Australia.
Acronym: FORTE '91
Sponsors: IFIP WG 6.1, IEEE, OTC and Telecom Australia.
Contact: Dr Ken Parker, Telecom Research Laboratories, PO Box 249, Clayton, Victoria 3168, Australia, Tel. +61 3 541 6797, Fax. +61 3 544 2362.
Email: k.parker@trl.oz.au.
Paper Submission Details:
Submission of full research papers (5 copies, 12 pt single spaced, max 16 pages) sent to: Prof Gordon Rose, Computer Science Dept., University of Queensland, Queensland 4072, Australia, Tel. +61 7 377 2766, Fax. +61 7 371 0783.
Email: rose@uqcspe.cs.uq.oz.au.

Date: November 26 - 30
Title: **Third Int'l Symp. on Parallel and Distributed Processing**
Location: Dallas, USA.
Sponsor: IEEE-CS.
Cosponsor: ACM.
Contact: Behrooz Shirazi, Univ. of Texas, Computer Science Eng. Dept., Box 19015, Arlington, TX 76019-0015; 'phone: (817) 273-3605, fax: (817) 273-2548.
Email: shirazi@evax.utarl.edu.

Date: December 3 - 5
Title: **The Fourth International Workshop on Petri Nets and Performance Models**
Acronym: PNPM 91.
Location: Melbourne, Australia.
Sponsor: Telecom Australia.
Contact: Jonathan Billington, Telecom Australia Research Laboratories, PO BOX 249, Clayton, Vic., 3168, Australia. Tel. +61-3-5416416, Fax. +61-3-5442362.
Email: j.billington@trl.oz.au.

Date: December 4 - 6
Title: **Software for Critical Systems**
Location: Fairmount Hotel, New Orleans, La, USA.
Acronym: SIGSOFT '91.
Sponsor: SIGSOFT.
Contact: Mark Moriconi, SRI International, 333 Ravenwood Ave., Menlo Park, CA 94025; 'phone (415) 859-5924.
Email: Moriconi@csl.sri.com.

Date: December 4 - 6
Title: **Int'l Conf. on Parallel and Distributed Information Systems**
Location: Miami Beach, Southern Florida, USA.
Sponsor: SIGARCH, SIGMOD, IEEE, CS, Florida International Univ.
Contact: Amit Sheth, Bellcore, 1J-210, 444 Hoes La., Piscataway, NJ 08854; 'phone: (908) 699-3300; fax: (908) 699-9011.
Email: amit@ctt.bellcore.com.
Paper Submission Details:
Submit seven copies of full paper by **May 10, 1991**, and abstract by electronic mail to H.V. Jagadish, 3C414A, AT&T Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974.
Email: jag@research.att.com.

Date: December 11 - 13
Title: **First International Workshop on Deontic Logic in Computer Science**
Location: Amsterdam, The Netherlands.
Acronym: DEON '91.
Contact: Dr. R.J. Wieringa, DEON '91, Department of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081HV Amsterdam, The Netherlands.
Paper Submission Details:
Authors are invited to submit five copies of their papers in English, double-spaced, not exceeding 5000 words, including a 10-line abstract and a list of keywords, addressed to the program chairperson, Prof. Dr. J.-J.Ch. Meyer by **June 1st, 1991**.

Date: December 17 - 19
Title: **Eleventh Conference on Foundations of Software Technology and Theoretical Computer Science**
Location: India International Centre, New Delhi, India.
Contact: H Saran (IIT Delhi), FST&TCS 11, Dept. of Computer Science Engineering, Indian Institute of Technology, Kanpur 208 016, India; 'phone: (0512) 244518/214151, Telex: 0325-296, 0326-392.

1992

Date: January 6 - 10
Title: **Formal Techniques in Real-time and Fault-tolerant Systems**
Location: University of Nijmegen, The Netherlands.
Contact: Prof. Dr. Ir. Jan Vytopil, Real-time Systems Group, Department of Informatics, University of Nijmegen, Toemoolveld, 6525 ED NIJMEGEN, The Netherlands; 'phone: +31-80-65 2075, Fax: +31-80-55 34 50, Telex: 48228 wina nl.
Email: vytopil@cs.kun.nl.

Date: Januray 8 - 10
Title: **The Fifth Refinement Workshop**
Location: London.
Sponsor: BCS-FACS.
Contact: Chairman: Professor Bernard Carré, Program Validation Limited, 26 Queen's Terrace, Southampton SO1 1BQ; 'phone: +44 (0) 703 330001.
Or: Local Organiser: Roger Shaw, Lloyd's Register, Lloyd's Register House, 29 Wellesley Road, Croydon CR0 2AJ; 'phone: +44 (0) 81 681 4818.
Email; rcs@uk.co.lreg.aie

Date: **April 6 - 10**
Title: **Latin American Theoretical INformatics**
Location: São Paulo, Brazil.
Acronym: latin '92.
Contact: Imre Simon, Program Committee Chair, latin '92 - Latin American Theoretical Informatics, Instituto de Matemática e Estatística, Universidade de São Paulo, Caixa Postal 20570, 01498 São Paulo, SP, Brasil; Fax: (55) (11) 815-4272.
Email: isimon@brusp.bitnet.
Or: Paulo Feofiloff, Organizing Committee Chair, latin '92 - Latin American Theoretical Informatics, Instituto de Matemática e Estatística, Universidade de São Paulo, Caixa Postal 20570, 01498 São Paulo, SP, Brasil; Fax: (55) (11) 815-4272.
Email: pfeofilo@brusp.bitnet.

Date: **April 20 - 23**
Title: **Computer Languages**
Location: San Francisco, California, USA
Sponsor: IEEE Computer Society
Co-Sponsors: ACM SIGPLAN & IFIP Working Group 2.4
Contact: James R. Cordy, Dept. of Computing and Information Science, Queen's University, Kingston, Canada K7L 3N6; 'phone: (613) 545-6054.
Email: cordy@qucis.queensu.ca
Or: Mario R. Barbacci, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA; 'phone: (412) 268-7704.
Email: barbacci@sei.cmu.edu
Paper Submission Details:
Submit 4 copies of papers or 3 copies of panel session proposals by *September 15, 1991* to Professor J.R. Cordy, programme committee chair, address above. Submissions should be accompanied by a cover letter that includes a return mailing address, telephone number and email address (if known). Authors will be notified of acceptance or rejection by *December 1, 1991*. Camera ready versions of accepted papers will be due *January 15, 1992*.

Date: **May 11 - 15**
Title: **14th Int'l Conf. on Software Eng.**
Acronym: ICSE 92
Location: Melbourne, Australia.
Cosponsors: IEEE Computer Soc. et al.
Paper Submission Details:
Submit six copies of full paper or experience report by **Sept. 6, 1991**, to A.Y. Montgomery, Computer Science Dept., Royal Melbourne Inst. of Tech., PO Box 2476V, Melbourne 3001, Victoria, Australia, 'phone 61 (3) 660-2943, fax 61 (3) 662-1617.
Email: aym@goanna.cs.rmit.oz.au.

Date: **May 27 - 29**
Title: **Assessment of Quality Software Development Tools**
Location: New Orleans, Louisiana, USA.
Sponsor: Tulane University.
In Coop.: IEEE Computer Society TCSE.
With Assist.: IBM Systems & Software Education
Contact: Ez Nahouraii, IBM (798/089), 6321 San Ignacio Avenue, San Jose, CA 95119, USA; Tel: (408) 281-5741.
Email: eznah@stlvm7.iinus1.ibm.com

Date: **June 29 - July 1**
Title: **ACM Conference on Lisp and Functional Programming**
Acronym: LFP '92
Location: San Francisco, California, USA.
Sponsor: SIGPLAN, SIGACT, SIGART.
Contact: Jon White, Lucid Inc., 707 Laurel St., Menlo Park, CA 94025, USA; 'phone: (415) 329-8400
Email: jonl@lucid.com.

Date: **January 19 - 22**
Title: **19th ACM Symposium on Principles of Programming Languages**
Acronym: POPL '92
Location: Albequerque, N. Mex., USA.
Sponsor: SIGACT, SIGPLAN.
Contact: Andrew Appel, Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08544-2087; 'phone: (609) 258-4627.
Email: appel@princeton.edu.

Date: **January 27 - 31**
Title: **The Second International Symposium on Environments and Tools for Ada**
Acronym: SETA2
Location: Washington DC, USA.
Co-Sponsor: ACM SIGAda, IEEE Computer Society TC on Computer Languages, Cooperation being sought with ACM SIGSOFT.
Contact: SETA2, c/o Prof. Gail E. Kaiser, Columbia University/Department of Comp. Sci., 500 West 120th Street, New York, NY 10027, USA.
Email: seta@cs.columbia.edu

Date: **January 28 - 31**
Title: **2nd International ACM/IEEE Symposium on Environments and Tools for Ada**
Location: Herndon, Va., USA.
Sponsor: SIGAda, SIGSOFT, IEEE TC CL.
Contact: Tricia Orbernborg, NADC Code 7031, Warminster, PA 18974-5000, USA; 'phone: (215) 441-2737.
Email: tricia@nadc.navy.mil.

Date: **February 5 - 7**
Title: **Harnessing the Object Revolution: Workshop on Object-oriented Software Engineering Practice**
Location: Denver, USA.
Contact: Dr. Pankaj Goyal, US WEST Advanced Technologies Inc., 4001 Discovery Drive, Boulder, CO 80303, USA; 'phone: (303) 541 6286, Fax: (303) 541 6300.
Email: pankaj@uswest.com, or: oows@uswest.com.

Date: **February 13 - 15**
Title: 9th Symposium on Theoretical Aspects of Computer Science
Location: Paris, France.
Acronym: STACS 92.
Sponsor: AFCET.
Co-Sponsor: GI
Contact: Prof. Alain Finkel, ENS Cachan - STACS 92, 61, avenue du Président Wilson, 94235 Cachan Cédex - France; 'phone: 33 1 47 40 22 74; fax: 33 1 47 40 20 74.
Email: finkel@enscachan.ens-cachan.fr

Date: **March 29 - April 2**
Title: **Software Engineering Tools and Techniques Workshop**
Location: New Orleans, LA, USA.
Contact: Steven Zilora, Creative Software Solutions, Inc., P.O. Box 192, Flanders, NJ 07836; 'phone: (201) 927- 8233; fax: (201) 927-7527.
Paper Submission Details:
Submit two copies of an abstract to Steven Zilora.

Date: **March 30 - April 1**
Title: **Eighth Int'l Conf. on Software Eng. for Telecommunication Systems and Services**
Location: Florence, Italy.
Acronym: SETSS 92.
Sponsor: Institution of Electrical Engineers.
Contact: IEE Conf. Services, Savoy Place, London WC2R 0BL, UK; 'phone: (071) 240-1871; fax: (071) 240-7735.

Date: July 8 - 10
Title: **International Symposium on Fault-Tolerant Computing**
Acronym: FTCS 22
Location: The Lafayette Hotel, Boston, MA, USA.
Sponsor: IEEE Computer Society, University of Massachusetts.
Contact: Prof. Dhiraj K. Pradhan, Conference Chairman, Electrical and Computer Engineering Dept., University of Massachusetts, Amherst, MA 01003; Tel: (413) 545-0160, Fax: (413) 545-4611.
Email: pradhan@ecs.umass.edu

Date: July 12 - 17
Title: **19th International Colloquium on Automata, Languages, and Programming**
Acronym: ICALP 92
Location: Technische Universität Wien, Austria.
Contact: Prof. Werner Kuich, Institut für Algebra und Diskrete Mathematik, Technische Universität Wien, Wiedner Haupstraße 8-10, A-1040 Wien, Tel.: +43 1 58801 5450.
Email: kuich@btx.UUCP.
Paper Submission Details:
Submit seven copies of an extended abstract to the Chairman of the Programme Committee, Prof. Werner Kuich before 15 November 1991.

Date: September 23 - 25
Title: **5th International Conference on Putting Into Practice Methods and Tools for Information System Design**
Location: Nantes, France.
Contact: Henri Habrias, Liana, IUT, 3 rue MI Joffre 44041 Nantes Cedex 01 (France); 'phone: (33) 40 30 50 56; fax: (33) 40 30 60 01.
Email: habrias@naiut.dnet@ciripa.circe.fr.
Paper Submission Details:
Submit five copies of their papers (15 pages maximum, double-spaced, (English or French) to Henri Habrias.