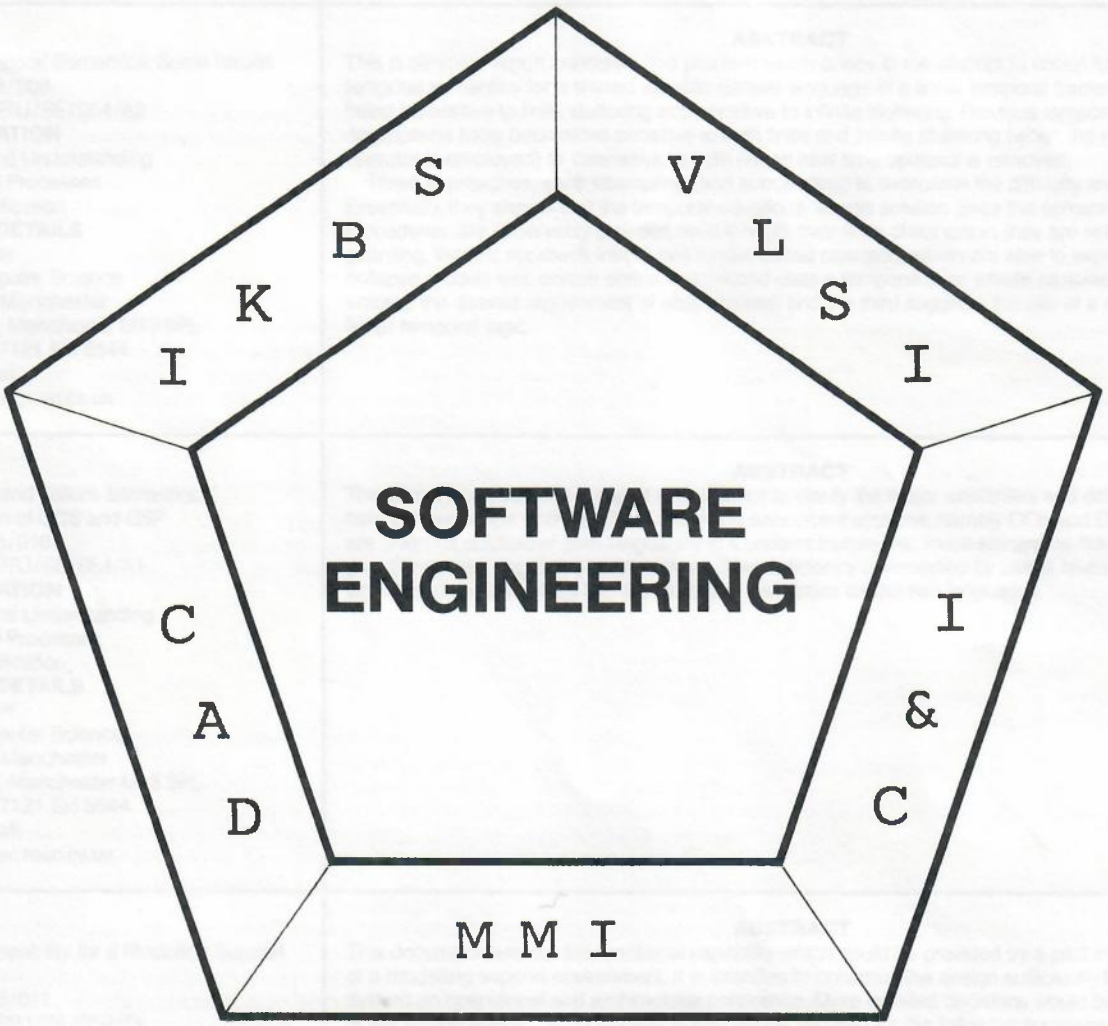
No. SE2/18

ALVEY NEWS SUPPLEMENT



ABSTRACTS OF COMPLETED DELIVERABLES FOR THE ALVEY SOFTWARE ENGINEERING DIVISION

FOR FURTHER INFORMATION ON THESE DELIVERABLES PLEASE REFER TO CONTACT DETAILS

TITLE
Implementation of Göteborg Type-Theory System on GEC-63 Series
REF: SE2/18/008
PROJECT: PRJ/SE/003
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Formal Specification
CONTACT DETAILS
R C Backhouse
Dept of Computer Science
University of Essex
Colchester CO4 3SQ
Tel: 0206 862286 Ext 2371

ABSTRACT
The implementation of Martin-Löf's Theory of Types developed by Kent Petersoon at Chalmers University of Technology, Göteborg, has been modified to execute on the GEC-63 Series. Rules for the subset type have been added to the implementation and several large scale exercises, including a proof of a while-rule, have been completed.

TITLE
Abstract Temporal Semantics: Some Issues
REF: SE2/18/009
PROJECT: PRJ/SE/054/A2
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Formal Specification
CONTACT DETAILS
Dr H Barringer
Dept of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL
Tel: 061 273 7121 Ext 5544
Electronic Mail:
howard@uk.ac.man.cs.ux

ABSTRACT
This preliminary report considers one problem which arises in the attempt to obtain fully abstract temporal semantics for a shared variable parallel language in a linear temporal framework, that of being insensitive to finite stuttering and sensitive to infinite stuttering. Previous temporal semantic descriptions have been either sensitive to both finite and infinite stuttering (when the next time operator is employed) or insensitive to both (when next time operator is removed).

Three approaches, each attempting (and succeeding) to overcome the difficulty are outlined. Essentially, they ensure that the temporal equations, whose solution gives the semantics of procedures, are observably guarded even though, over finite observation, they are not! To enforce guarding, the first approach introduces model-based operators which are able to expand and collapse models w.r.t. certain states, the second uses a temporal logic whose canonical models enforce the desired requirement of abstractness, and the third suggests the use of a dense-time linear temporal logic.

TITLE
Acceptance and Failure Semantics: A comparison of CCS and CSP
REF: SE2/18/010
PROJECT: PRJ/SE/054/A1
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Formal Specification
CONTACT DETAILS
Dr H Barringer
Dept of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL
Tel: 061 273 7121 Ext 5544
Electronic Mail:
howard@uk.ac.man.cs.ux

ABSTRACT
The primary purpose of this report is to attempt to clarify the major similarities and differences between two major approaches to modelling concurrent systems, namely CCS and CSP. Semantics are given for subsets of both languages in a uniform framework; these semantics, however, do not give the desired degrees of abstractness. This deficiency is remedied by use of blurring functions which map our semantics into the 'standard' semantics for the two languages.

TITLE
Functional Capability for a Modelling Support Environment
REF: SE2/18/011
PROJECT: PRJ/SE/059/DI
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Formal Specification
CONTACT DETAILS
C Harvey
Project Leader
British Telecom Research
Anzani House, Trinity Avenue
Felixstowe IP11 8XB
Tel: 0394 697510

ABSTRACT
This document sets out the functional capability which could be provided by a pilot implementation of a modelling support environment. It is intended to constrain the design sufficiently to give the system an operational and architectural coherence. More detailed decisions would be the province of the design teams. The document is intended to be used for the following purposes:

i) to provide a starting point for the detailed design of modelling tools and the support system.
ii) to provide a starting point for the detailed implementation plan.
iii) to delineate the world-view of the user and provide input for an evaluation strategy.

The performance modelling environment described provides an extensible set of applications tools covering the areas of performance specification, model construction, experimentation and reporting, according to the stated design requirements. The tools and work-in-hand are manipulated via a uniform style of user-driven interface which uses the desktop metaphor. To define a pilot capability which is achievable in the required timescale, it will be necessary to give some aspects priority over others. In doing this, a balance has to be struck between two conflicting objectives: to provide an investigative prototype for an advanced design while at the same time providing a basic capability that can be separately exploited by the respective partners in the shorter term. These considerations have led to the following principles for pilot design and development.

There should be a comprehensive conceptual model which is based on a fairly ambitious view and not constrained by resourcing or timescales.

Features which are to be given priority should be chosen from two categories: 'safe' and 'innovative'.

The specification tools and some features of the process-interaction tools are regarded as innovative. The experimenter tool, while novel, need not impose great strains on the state of the art, provided a safe subset of the possible features is chosen. The experimenter can perhaps be thought of as a natural spin-off from the concept of an integrated environment. Reporting is likely to be more primitive than we would wish, because of the absence of adequate UNIX-based document processing facilities. The queue network tool represents the safest option for exploiting a graphical user interface, irrespective of which solvers lie behind it or whether they will use analytic or simulation methods.

TITLE
The Development of a System for Concurrent Device Verification
REF: SE2/18/012
PROJECT: PRJ/SE/018
CLASSIFICATION
Integration and Implementation
Methods and Processes
Quality Assurance
CONTACT DETAILS
Dr G J Milne
Dept of Computer Science
University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ
Tel: 031 667 1081 Ext 2709
Electronic Mail Address:
GM@ECSVAX

ABSTRACT

This research has resulted in the discovery and development of rigorous techniques for concurrent system analysis. While these techniques apply to concurrent software systems they have been developed with VLSI applications in mind.

A prototype functional simulator has been implemented which directly "runs" descriptions of circuit behaviour expressed in the CIRCAL language. Also developed are a number of techniques for verifying design correctness by proof: hierarchical verification that a design meets its specification and the concept of verification within restricted contexts.

These techniques are seen as being crucial to the task of validating inherently complex systems, both hardware and software. They aim to tackle the issue of descriptive complexity head on and coupled with a hierarchical design methodology they lay the basis for an integrated design and verification system, work that is continuing under Alvey Grant VLSI-CAD/011.

TITLE
The Project Development Environment (PDE)
REF: SE2/18/013
PROJECT: PRJ/927/1
CLASSIFICATION
Implementation and Integration
Methods and Processes
Environment
CONTACT DETAILS
Prof M K Crowe
Software Tools Research Group
Department of Computing Science
Paisley College of Technology
High Street, Paisley
Tel: 041-887 1241

ABSTRACT

The PDE is a tool designed to support Software Configuration Management under the UNIX* operating system. It is sufficiently flexible to allow project development methodologies and objectives to vary from one project to another.

Among the tools and facilities offered are:

- All the standard UNIX commands
- An on-line archiving system
- A version control mechanism
- Attributed object management
- Controlled access facilities

The PDE was developed, by the STRG, during the first stage of a three year SERC-funded research project at Paisley College, and is available under licence from NRDC, together with a minimal project schema. The project, which is not due to finish until September 86, is called "An Incremental Compiler for Ada with Enhanced Software Quality Assurance Functions".

* UNIX is a trademark of Bell Laboratories

TITLE
The York UNIX-X25 Communications Package
REF: SE2/18/014
PROJECT: PRJ/932/4
CONTACT DETAILS
Martin Gray
Spider Systems Limited
65 Bonnington Road
Edinburgh EH6 5JQ

ABSTRACT

The York UNIX-X25 package enables a host computer running the UNIX operating system to connect to X25-based networks such as JANET and BT's SwitchStream One. The basic facilities offered are:

- terminal access to the host from other hosts;
- terminal access to other hosts from the host;
- sending of files and mail to other hosts;
- receipt of files and mail from other hosts.

There are sixteen X25 logical channels to the network. These are shared by the four services listed above according to the requirements of each site.

In addition, there are utility programs to control access to networks, to collect and analyse usage statistics and to maintain administrative files.

In order to avoid loading the host CPU with the extra processing necessary to maintain a network link, the system is functionally split between the host and a small front-end computer. The front-end is usually based on a Falcon SBC-11/21. It supports a single serial synchronous line to the network and performs much of the protocol work.

The front-end may be connected to the host by a synchronous link or an asynchronous one. The synchronous link is faster and far less prone to transmission errors.

The host requires a suitable driver to be integrated into the kernel to support the link.

User documentation for all host programs is supplied as manual pages. System documentation, intended for system programmers and system managers, is also provided; it includes Hardware and Software Installation Guides.

TITLE
MAJIC—A Computer Aided Program Design for the Commercial Computer
REF: SE2/18/015
PROJECT: PRJ/SE/913/938
CONTACT DETAILS
Dr A G Sutcliffe
Department of Computation
UMIST
PO Box 88, Sackville Street
Manchester M60 1QD
Tel: 061 236 3311

ABSTRACT

MAJIC (M.A. Jackson Interactive Computer aided program design) is a program development tool which supports the Jackson Structured Programming method. The principle objective of MAJIC is that program design and coding should be automated by code generation directly from specifications; thus development and maintenance of programs is facilitated and users can concentrate on the more creative aspects of program design.

The JSP technique ensures that program structures are correctly derived from the data structures which they process. The method has a series of well defined steps and algorithms to guide the designer and is therefore well suited to automation. Additionally it is a practical program design method which has been commercially practiced for several years.

The project deliverables are as follows:

- A data structure diagram editor for creating diagrams and inserting, deleting and changing components.
- A program structure generator which allows users to build program structures from data structures following the JSP rules of correspondence.
- A program structure modifier which implements the JSP design refinements according to Backtracking and Inversion.
- An Actions and Conditions syntax checker/editor which enables the user to add elementary operations to the specification and subsequently add COBOL statements for actions and conditions.
- Automatic code generation in ANSI 74 COBOL from diagram specifications with the minimum addition of code for actions and conditions.

- A data dictionary storing diagram structures and other parts of the specification. This allows libraries of data structures to be built up.

The whole of MAJIC has been completed and tested except for the syntax checker part which is near completion.

Program specification is carried out by building diagrams and then adding pre-validated code for detailed program operations. Prevalidation is achieved by either presenting users with picking lists which contain only valid constructs for each step, or only allowing them to carry out valid (in JSP terms) operations on a specification. In this way the maximum accuracy is ensured within the limitations of JSP's formalism. The system generates all structure related code directly from diagram specifications making program development more efficient and bringing the benefits of the JSP method into an easy to use development environment.

TITLE
Essential Mathematics for Software Engineers
REF: SE2/18/016
PROJECT: PRJ/SE/042/03
CLASSIFICATION
Exploitation and Evaluation
Methods and Processes
Education and Formal Specification
CONTACT DETAILS
Dr I D Huntley
Dept of Mathematical Sciences
Sheffield City Polytechnic
Pond Street, Sheffield, S1 1WB
Tel: 0742 20911 Ext 2204

ABSTRACT

Formal specifications provide a basis for rapid prototyping of software to demonstrate the fundamental features to users via a prototype rather than via the original paper-based specification. Formal methods utilise the concepts of discrete mathematics to build a model of a system through which solutions to specific problems can be proved and evaluated in a cost effective way.

Essential Mathematics for Software Engineers is an integrated text and video pack which teaches this field of mathematics and shows how it is applied to software problems. It provides a foundation upon which a software engineer can build expertise in the use of formal specification languages.

The course has been designed and written to be studied either as a taught course of about 30 hours duration or as a self-learning course where students work on their own, at their own pace, and in their own time. In addition to the published text and access to a video player, the student requires a tutor who has some familiarity with discrete mathematics.

TITLE
Advanced Support Environment Study Final Report
REF: SE2/18/017
PROJECT: PRJ/SE/002/2
CLASSIFICATION
Implementation and Integration
Management, Environment
IPSE2, IPSE3
CONTACT DETAILS
Mr M K Tordoff
ICL
West Avenue, Kidsgrove
Stoke-on-Trent ST7 1TL
Tel: 0782 29681

ABSTRACT

This report is from the study group established to clarify the characteristics required and expected of advanced support environments for the software engineering industry, paying particular attention to what is attainable in the 5 and 10 year timescales.

The Alvey Software Engineering strategy document characterised the development of IPSEs in broad terms. The study investigated the more specific requirements of IPSEs, the technological or other advances which will influence IPSEs, and the capabilities likely to be provided by successive generations of IPSEs. Information was gathered from the information systems industry at large, supplemented by detailed experiences and projections from individuals, where appropriate.

The document reports the conclusions and findings of the study regarding:

1) The variety of requirements which the information systems industry has of advanced support environments.
2) The analysis of these requirements to identify key issues.
3) The identification of technological trends and aspirations relating to advanced support environments, and the indications these give to the direction work on such environments should take.
4) Detailed recommendations for projects which should be done.

TITLE
Visit to the Universities of Bath and Edinburgh by Professor C S Tang
REF: SE2/18/018
PROJECT: PRJ/SE/061
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Education
CONTACT DETAILS
Dr P J L Wallis
School of Mathematics
University of Bath
Claverton Down, Bath BA2 7AY
Tel: 0225 61244

ABSTRACT

Professor Tang of Academica Sinica, Beijing, visited the University of Bath from September 1985 to January 1986 with two weeks at the University of Edinburgh in December 1985. He also paid short visits to other UK universities and industrial establishments, gave a number of lectures and seminars, and undertook some personal research. His lectures and seminars all described various aspects of the XYZ system he is developing in Beijing, which provides a unified description method for many different aspects of computer science based on the use of temporal logic.

TITLE
Ponder Compiler
REF: SE2/18/019
PROJECT: PRJ/SE/006
CLASSIFICATION
Implementation and Integration
Methods and Processes
Software Generation
CONTACT DETAILS
Dr Jon Fairbairn
University of Cambridge
Computer Laboratory
Corn Exchange Street
Cambridge CB2 3QG
Tel: 0223 334600
Electronic Mail: jf@UK.AC.Cam.CL

ABSTRACT

The latest release of the Ponder compiler (4.29, 8.07) for VAX Unix is now available. This corrects some mistakes in earlier versions and the front end is somewhat faster than version 4.27.

Ponder is a pure applicative programming language based directly on the lambda-calculus. It was developed for experiments in functional programming style, but has proved suitable for general purpose functional programming. The language is designed with very few primitive constructs, but contains mechanisms that allow more powerful syntactic forms to be built upon the small kernel. Of particular interest is the polymorphic type system, which allows functions that require polymorphic arguments.

The implementation uses strictness detection and supercombinators to compile into VAX machine code, producing functional programmes that are among the fastest available. The performance for first order strict function is comparable to conventional high level languages.

TITLE
Development of a Programming Calculus
REF: SE2/18/020
PROJECT: PRJ/SE/022
CLASSIFICATION:
Innovation and Understanding
Methods and Processes
Formal Specification
CONTACT DETAILS
Dr R S Bird
Programming Research Group
University of Oxford
8-11 Keble Road, Oxford
Tel: 0865 54141

ABSTRACT

The project aimed to develop concepts and notations suitable for a mathematical style of program transformations, to realise these concepts in a functional language based on recursion equations and definitions of higher-order operators, and to investigate requirements of a computer aided system for carrying out transformations expressed as laws about operators.

Notations and concepts have been investigated and the functional language, designed in collaboration with Dr P Wadler (of the PRG) was implemented on a Vax 11/780 (first under VMS, later under UNIX) rather than on SAGE microcomputer as the latter proved insufficient for the purpose. The language ('ORWELL') has been ported to SUNs and will be used as a teaching language for first year undergraduates in the Mathematics and Computations Honours School at Oxford.

TITLE
Software Engineering Conferences and Travel 1985
REF: SE2/16/021
PROJECT: PRJ/SE/038/1b
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Education
CONTACT DETAILS
Dr P J L Wallis
School of Mathematics
University of Bath
Claverton Down, Bath BA2 7AY
Tel: 0225 61244 Ext 216

ABSTRACT

This small grant provided travel support for Dr Wallis' software engineering and Ada-related activities during 1985.

It financed attendance at the Ada UK Conferences (York, January 1985), Tapsoft (Berlin, March 1985), and the Ada-Europe Conference (Paris, May 1985), and also funded some attendance at meetings of Ada-Europe Working Groups.

Reports on the Tapsoft and Ada-Europe 1985 Conferences have appeared in the Alvey Software Engineering Mailshot; attendance at meetings of the Ada-Europe Numerics Working Group resulted in a contribution by Dr Wallis to Scientific Ada, a book by members of the Working Group to be published during 1986.

TITLE
Z Handbook
REF: SE2/18/022
PROJECT: PRJ/SE/065/01
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Formal Specification
CONTACT DETAILS
Mr N Haigh
RACAL ITD
Richmond Court
309 Fleet Road, Fleet
Hampshire GU13 8BU
Tel: 0252 622144

ABSTRACT

The specification notation Z has evolved a good deal since its introduction in 1979. Apart from course material and certain monographs and other papers describing specifications written in Z, there has not been a generally available description of the notation. FORSITE and other current industrial projects require a definitive handbook providing a description upon which automated tools, such as time checkers, can be built. The PRG at Oxford University, through the FORSITE project, have delivered a first draft of such a handbook. This handbook contains a description of:

1) The notation
2) The underlying mathematics
3) The schema notation
4) The type system
5) The rules of reasoning

The handbook will include first published drafts of the Z type system and rules of reasoning. The draft is issued for two purposes. Firstly, to meet the need for a cohesive document describing Z, secondly, to obtain feedback from readers which will improve the quality and content.

TITLE
MMI for Eclipse Version 1 Functional Specification
REF: SE2/18/023
PROJECT: PRJ/SE/014/10
CLASSIFICATION
Innovation and Understanding
Methods and Processes
ISPE 1
CONTACT DETAILS
A E Elliston
Software Sciences Ltd
Park Street, Macclesfield
Cheshire
Tel: 0625 29241

ABSTRACT

In ECLIPSE Version 1 the end-user will utilise a workstation interface. The workstation chosen is the SUN2 monochrome range running VAX 4.2 bsd. The database environment runs on a VAX based on UNIX 4.2 bsd. The workstation interface will enable a significant amount of processing to be carried out remotely from the VAX. The communication medium is Ethernet.

The MMI facilities of ECLIPSE will access and be accessed by other components of ECLIPSE. Both the MMI and application code is dependent upon the facilities of the ECLIPSE ITI (Integrated Tools Interface).

To provide the flexibility and modifiability demanded of it, the MMI offers toolworkers a set of tool fragments which carry out specific tasks. These fragments include text processing, graphics processing, menu management, forms management, command and environment management, help and error message management, screen and window printing.

TITLE
ECLIPSE Version 1 Integrated Tools Interface Specification
REF: SE2/18/024
PROJECT: PRJ/SE/014/64
CLASSIFICATION
Innovation and Understanding
Methods and Processes
Environment
IPSE 1
CONTACT DETAILS
A E Elliston
Software Sciences Ltd
Park Street, Macclesfield
Cheshire
Tel: 0625 29241

ABSTRACT

This document specifies that component of ECLIPSE VI known as the Integrated Tools Interface for Version 1 (ITI-V1)

In the ECLIPSE two-tier database versioned data is stored in a number of versioned units, each of which comprise:

- an object in the first tier
- optionally, an object body in the second tier. The object body may contain data whose structure is defined by an IDLE (Interface Definition Language for ECLIPSE) schema. An object body is stored as a filestore file.

In the two-tier database, an object body is attached to an object and can only be accessed through control information recorded in that object.

In ECLIPSE Version 1, there is no direct interface between ITI and the two-tier database. Data is stored and retrieved from the database by import and export operations.

Export is a user invoked function which returns a structured filestore file. The file is an exact copy of the body of the nominated object.

Import is a user invoked function which takes a local filestore file and stores it in the ECLIPSE

database. The file is to be stored as the new version of the nominated item. In order to achieve this, import creates a new object in the first tier.

ITI-V1 enables update access to be obtained, the contents of a structured body. Any number of bodies may be open at the same time (subject to implementation constraints). However, each body is disjoint from any other since reference between bodies are not supported.

ITI-V1 can only be used to interpret the data held in a body whose structure is defined by IDLE.

Full ITI may be regarded as providing a unified interface to the following components of ECLIPSE:

- SDS2 database management system
- Foundation object management system
- Interface Definition Language for ECLIPSE

ITI V1 is a limited subset of the above, and provides that part of the full interface which maps onto the IDLE component.

TITLE
Development of a Strategy for Technology Transfer in Relation to the FOREST Project
REF: SE2/18/025
PROJECT: PRJ/SE/015/R11
CLASSIFICATION:
Innovation and Understanding
Methods and Processes
Education
CONTACT DETAILS
Mr E A Pacello
GEC Research
West Hanningfield Road
Great Baddow, Chelmsford
Essex CM2 8HN
Tel: 0245 73331

ABSTRACT

The report describes research into problems which will be encountered when introducing into industry formal mathematically based methods such as those developed in FOREST. The analysis provides the background against which a strategy for the transfer of FOREST into industrial use will be developed.

The report includes:

- a model of the technology transfer process
- project mission: the objectives of FOREST
- opportunity analysis: the potential market place
- user behaviour: cognitive aspects and the psychology of learning a formal system
- organisational aspects of technology transfer
- the influences of the environment
- the role of formal methods in system production
- conclusions

TITLE
MOS Simulation in HOPE
REF: SE2/18/026
PROJECT: PRJ/SE/046/1
CLASSIFICATION:
Innovation and Understanding
Methods and Processes
Software Generation
CONTACT DETAILS
Dr R Offen
Imperial Software Technology
60 Albert Court, Prince Consort Road
London SW7 2BH
Tel: 01-581 8155

ABSTRACT

This report describes the development of a MOS circuit simulator written in HOPE, a pure functional language. An earlier report described how the main simulation function was derived by applying intuitive transformations to improve the efficiency of a first highly recursive program. This report is concerned with the development of a useable "tool" from the core function.

One of the main additions to the program has been the provision of a facility to process a sequence of inputs to the circuit. Circuits are constructed in a hierarchical manner from smaller components. For example, a one-bit adder is composed from two half bit adders which in turn are a collection of nand gates and inverters. This led naturally to a tree data structure for holding the internal signal values of the circuit. For any but the most trivial problem, this is a more efficient solution than a simple list.

The user interface has been extended to provide selective output to the terminal whereas the circuit description is read in from a file. A second file lists some parameters which give the terminating conditions for the program, for example, "stop when the following signal values have been reached".

TITLE
HOPE-based JSD
REF: SE2/18/027
PROJECT: PRJ/SE/046/2
CLASSIFICATION:
Innovation and Understanding
Methods and Processes
Software Generation
CONTACT DETAILS
Dr R Offen
Imperial Software Technology
60 Albert Court, Prince Consort Road
London SW7 2BH
Tel: 01-581 8155

ABSTRACT

This report examines the use of Jackson System Development (JSD) as the basis of a method for developing functional programs in HOPE. The "intermediate" representations of the JSD sequence are based on networks of communicating processes. These can be represented by recursive functions that map lists of input messages to lists of output messages.

The Implementation Step involves (1) scheduling processes so as to run on the available machine resources, and (2) providing a user interface.

The JSD process functions are made non-recursive, and are explicitly activated by a Scheduler process. In effect, a suspend-and-resume mechanism is incorporated into the JSD processes. Control is passed between processes when messages are written. The Scheduler maps a single stream of "external" input messages to a single stream of "external" output messages.

The input messages are generated by a process that parses a stream of characters from a terminal. This process also generates prompt and error messages for immediate output. A third process merges messages from the Scheduler (ie the JSD system) and the Parser (ie the user interface), converts the messages to character strings, and displays them. These three processes (like the JSD processes) are inherently concurrent, but since our target is a sequential system, these processes must be explicitly activated (one at a time) by a Super-scheduler. This allows an interactive sequential system to be generated from a JSD Function Step representation.