ISSN 0254–2757

QI QUÆSTIONES INFORMATICÆ

Volume 5 • Number 3 December 1987

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereeneging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

An official publication of the Computer Society of South Africa
and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa
en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

Annual subscription are as follows:

	SA	US	UK
Individuals	R10	$ 7	£ 5
Institutions	R15	$14	£10

Computer Society of South Africa
Box 1714 Halfway House

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

ABSTRACTS OF OLD MUTUAL CONFERENCE FOR PH.D AND M.SC STUDENTS IN COMPUTER SCIENCE, STELLENBOSCH, 2-3 October 1987

THE KERNEL OF A MULTITASKING μ-OPERATING SYSTEM

M.C. Ackerman, *University of Stellenbosch*

The Client-Server model imposes on operating systems a structure which provides conceptually simple distributability. In this model the operating system consists of a minimal kernel which provides a process management and communication environment on which all other operating system tasks run as user processes.

The kernel consists of a layer which provides the mechanics of process management and interprocess communication (IPC), and a layer which contains a driver for each IO device. IO tasks communicate with each other and user processes only through the message passing facilities provided by the lowest layer of the kernel.

An example of an implementation of this model is the MINIX operating system which is coded in C. Concurrent process creation and IPC are not primitives of the C language but are provided by system calls (Fork and Exit) and library routines (Send and Receive). Since the scope rules of C cause default visibility of variables this language does not conform closely to the Client-Server model.

Communicating Sequential Processes (CSP) is widely in use as a notation for the specfication of operating systems, yet not one operating system coded in a CSP implementation has been done. The process and communication concepts of CSP has been merged with a minimal Pascal subset resulting in the new concurrent language, Joyce.

An implementation of a Joyce compiler is currently under way, and an operating system based on the Client-Server model and MINIX will subsequently be implemented in Joyce. That operating system will then be distributed on a multiprocesspor network.

SPS-ALGOL : SEMANTIC DATA MODEL CONSTRUCTS IN PS-ALGOL

S. Berman, *University of Cape Town*

Persistent data is defined as that which survives on secondary store after program execution terminates. In a persistent programming language there is no distinction whatever between persistent data and other objects. In contrast to conventional systems, both the data structures and the operations applicable to "permanent" data are identical to those used for short-term data.

PS-Algol is the only implemented persistent language at present. This paper describes a proposed extension to this language, SPS-Algol, in which concepts and tools normally associated with databases and semantic data models are incorporated. In particular transactions, type hierarchies, association, data derivation, semantic integrity constraints and subschemas are proposed as useful extensions to the programming language, and some of the implementation issues are discussed. Finally, some outstanding research problems in the field of persistent languages are briefly described to indicate directions for future work.

COLOUR IMAGE QUANTIZATION USING GRID FILES

R.F. Breedt

Colour image quantization is used to display high-fidelity reproductions of images on small frame buffers. Colour quantization can be broken into two phases :

1. Computing a colour map based on colour statistics from the original image.
2. Mapping the original colours to their nearest neighbours in the colour map.

The grid file is an adaptable multikey file structure that achieves an upper bound of two disk accesses for single record retrieval and handles range queries efficiently. An efficient method for adaptive, tapered quantization of colour images, using a grid file for both phases, is presented.

A RELATIONAL THEORY OF COMPUTATION

K. Britz, *University of Stellenbosch*

A terminating deterministic program is a function which is applied to input objects in order to yield output objects. For each input object there is a unique output object. A generalization of this concept is to regard non-deterministic programs as relations. A program is applied to a set of input objects. For each input object there may be more than one output object. This idea is formalized in a relational theory of computation, which is based on a first order logic of relations. The behaviour of relations and operations on relations are axiomatized in the logic, providing the necessary framework to develop a comprehensive theory of computation. This elegant mathematical basis simplifies the study of the semantics of the language, yet is rich enough to act as programming language proper and faciltates proofs of properties such as the equivalence of programs.

REAL TIME EXPERT SYSTEMS

D. Bryant, *University of Stellenbosch*

The increasing popularity of real time systems has resulted in a need for highly complex control and analytical systems, partly due to the inability of humans to perform consistently in high pressure situations.

Real time systems are characterised by critical time constraints and a small margin for error. Problems arise in real time process control when user interaction is required by the system. A user may be required to supply parameters, supplement process information or make a decision choice. It is in the latter situations, where the user has to take an irrevocable decision under pressure that many of the limitations of humans become apparent, ie their tendency to overlook relevant data, to respond both inconsistently and slowly and their tendency to panic when the rate of information flow becomes too great. Real time expert systems are an attempt to overcome these shortcomings.

An expert system is a computer application that uses explicitly represented knowledge, logical deduction and computational inference techniques to achieve a level of performance comparable to that of a human expert in some applicable area or domain.

EXPLANATION IN EXPERT SYSTEMS

K.J. Carden, *Rhodes University*

The ability of an expert sytem to explain its reasoning is fundamental to the system's credibility. Explanations become even more vital in systems which use uncertainty.

The research described here has been conducted into current explanation facilities, as well as the needs of some local expert systems users. The results of this have lead to the development of a powerful explanation sub-system. This sub-system interfaces with an existing production rule-based expert system shell which incorporates a method of handling uncertainty.

A number of the features implemented have been based solely on the identified needs of the local expert systems and have not been noted in the literature or in other systems.

The shell and explanation sub-system have been developed in Turbo Prolog.

PROGRAMMING LANGUAGE CONSTRUCTS FOR INTERRUPT-GENERATING MEMORY

P.G. Clayton, *Rhodes University*

Interrupt-generating memory cells are suggested as a means for creating opportunities for concurrent program execution, and as a feature for computers which simulate human behaviour rather better than current Von Neumann designs do. Although such architectures are not currently available, multi-processor computers can be used to simulate machines which use interrupt-generating memory.

This paper discusses ways in which a traditional procedural programming langauge can be implemented to exploit multi-processor environments or computer architectures which are designed around interrupt-generating memory cells. The paper concentrates on techniques which enable such a programming language to be implemented without resorting to non-standard source language statements or directives, so that the precise nature of the computer architecture on which the application program will be executed remains invisible to the high level programmer.

EXPERIENCES IN DRAFTING AN EXCHANGE STANDARD

A.K Cooper, *University of Pretoria*

Geographically referenced (geo-referenced) information consists of all information that refers to the man-environment system and that can be localised in space and time. With the advent of computerisation, large volumes of geo-referenced information are being captured in a digital form. Due to the costs and time involved in capturing geo-referenced information, it is desirable to have a standard through which users can exchange geo-referenced information.

The author is a member of a project team drawing up a proposed national standard for the exchange of digital geo-refererenced information. His particular responsibilty is the data structure of the exchange standard, which forms part of his M.Sc. thesis. The final version of the exchange standard will be released to the public during September 1987. This paper presents some of the author's experiences in drawing up the proposed exchange standard.

A SYSTEM DESIGN METHODOLOGY FOR COMPLEX ELECTRONIC SPREADSHEET MODELS

N.L.O. Cowley, *University of Port Elizabeth*

The issues involved in designing and implementing electronic spreadsheet models which achieve reasonably ambitious goals are explored. This is seen as important in view of the growth in the numbers and computer competence (or computeracy) of end users, and the ubiquitous nature of electronic spreadsheets.

It is important to apply a systematic and planned approach to worksheet development. A modification of traditional systems analysis and design or the newer method of prototyping both lend themselves to such an approach. Particular implementation problems posed by the nature of the spreadsheet environment, and a set of tools and techniques to simplify implementation, are discussed.

A large and reasonably complex spreadsheet model was constructed as part of the research, and four more are planned. Lessons learned from this model are discussed.

THE PRACTICAL SIDE OF VERIFICATION

P.J.A. de Villiers, *University of Cape Town*

Program verification is about 20 years old now. The goal is to make programs more reliable, but in practice program verification is seldom used. The reason may be that many researchers seem to be more interested in producing the "ultimate" verifcation technique than in applying existing techniques in practice. Some kinds of programs are better suited to verification than others and these should be concentrated on. A project to investigate the use of mechanical verification techniques to verify concurrent programs is in its initial stages and will be discussed.

PERSOONSIDENTIFIKASIE DEUR MIDDEL VAN NAAMPASSING IN 'N GENEOLOGIESE DATABASIS

C. du Plessis, *Universiteit van Port Elizabeth*

Huidiglik word inligting oor mense wat sedert 1652 in Suid-Afrika gewoon het, in 'n geneologiese databasis gelaai. Deur die jare het dit egter gebeur dat die spelling van sommige families se vanne verander het. A.g.v. onvolledige en onbetroubare data is daar besluit dat wanneer 'n persoon se inligting gelaai moet word, 'n stelsel moet bepaal watter vanne in die databasis soortgelyk aan die persoon se van is.

Daar bestaan verskillende algoritmes wat kan bepaal in hoe 'n mate vanne soortgelyk is. 'n Paar van hierdie algoritmes word bespreek. Die eerste groep algoritmes bepaal 'n kode as voorstelling van 'n van. Alle vanne met dieselfde kode word dan as soortgelyke vanne beskou. Hierdie metodes neem dus nie in ag dat sekere vanne meer soortgelyk is as ander nie.

Ander, meer komplekse, algoritmes bepaal 'n afstandsmaat tussen vanne. Deur hiervan gebruik te maak kan alle vanne wat soortgelyk aan 'n spesifikasie van is, gerangeer word.

Vir elke algoritme word die voordele en nadele daarvan verbonde uitgelig. Verder word moontlike aanpassings met hul voordele uitgewys.

Die bespreking word afgesluit met 'n verwysing na die finale mikpunt, nl. die daarstelling van 'n stel reels wat gebruik kan word om te bepaal watter vanne soortgelyk aan 'n gegewe van is.

DIE ONTWIKKELING VAN 'N OUTOMATIESE PROGRAMMERINGSTELSEL

J. P. Du Plessis, *Universiteit van die Oranje Vrystaaat*

Outomatiese programering behels die outomatisering van een of meer van die verskillende fases van programmering. Daar is verkeie benaderings tot Outomatiese Programering. EXPROG is so 'n outomatiese programmeringstelsel wat van die kennisgebaseerdebenadering gebruik maak. Invoer tot EXPROG is 'n probleemdefinisie met behulp van enkelvoudige sinne in Engels. EXPROG konstrueer 'n algoritme en lewer dan as afvoer 'n Pascal-program wat die gevraagde probleem oplos. EXPROG los probleme, wat in die elementere wetenskaplike gebied voorkom, op. 'n Geskikte voorstellingsmetode moet ontwikkel word.

SIMBOLIESE WISKUNDE IN 'N PRODUKSIESTELSEEL OMGEWING

A. J. Du Plooy, *Universiteit van die Oranje Vrystaaat*

LEEK (Leer Ekspert) maak gebruik van die produksiestelsel voorstellingsmetode, aangesien dit 'n redelike eenvoudige benadering (miskien ook die gewildste) in Ekspertstelsels is. Aangesien die leeraspek hier belangrik is, is 'n eenvoudige voorstellingstipe baie nuttig sodat daar dan meer op die "Leer" in plaas van die voorstelling gekonsentreer kan word.

Daar bestaan verskeie benaderings tot "Leer" in ekspertstelsels, nl., leer volgens analogie, leer deur foute, leer deur induksie, leer deur eksplorasie en leer deur vertel te word en nog ander variasies. In hierdie benaderings is daar heelwat oorvleueling; dit sal dus interessant wees on soveel moontlik van die eienskappe in 'n ekspertstelsel saam te vat. LEEK word tans ontwikkel om hierdie aspekte na te vors.

COMPUTER HYPHENATION OF AFRIKAANS

Quintin Gee, *University of the Witwatersrand*

The hyphenation of words in a language is a relatively easy process for the mother-tongue typist since a word can be enunciated syllable by syllable, and these provide "natural" break points. However, a computer can only carry out a set of instructions and has no understanding of the meaning of the words or their pronounciation. All it can operate on is a string of symbols. This means that hyphenation rules such as "Break between meaningful components" are not very useful, although linguistically accurate. An example of a rule more acceptable for the computer would be "Always split doubled consonants", but inevitably there are exceptions to this.

There is a need for South African publishers and printers to correctly hyphenate many languages. In particular, the research to achieve correct and efficient hyphenation of Afrikaans has not yet been properly addressed. As the spread of Personal Computers becomes wider, the problem of handling text correctly intensifies. Most Word Processors on sale in this coutry ignore both accents and hyphenation.

One approach is that of storing a whole Afrikaans dictionary containing the hyphens already positioned manually. This inevitably ignores the problem of creating new compound words and thus cannot be said to be a general solution, although it will satisfy the requirements of all common needs.

For small computers, this is too expensive an approach to undertake currently, and an algorithmic solution is preferable, relying on a dictionary to contain idiosyncrasies such as imported words and proper names.

This paper describes some of the orthographic formation of syllables in Afrikaans and shows how the syllables can be analysed to determine how to recognise syllable breaks. This gives rise to several data representation problems.

TOWARDS AN EGGSPERT

E. Goedeke, *University of Natal*

The objective of the Masters is to develop and implement and Expert System on the hatchability and fertility of chicken eggs, hence the name Eggspert. The problem domain falls into the diagnosis category of Expert Systems. The talk will concentrate on the knowledge representation, inference engine and treatment of uncertainty used in the expert system building tool used to implement Eggspert. Most of the building tool is still in the design stage and needs to be refined before being implemented.

IMPLEMENTING OCCAM ON A DISTRBUTED SYSTEM

D.T. Hill, *Rhodes University*

Occam is designed for concurrent programming on a network of transputers. The language contains some restrictive configuration constructs used in distributing a program across a network. An alternative approach is proposed which abstracts the occam program from the transputer network configuration. Problems are presented, and solutions discussed for an implementation on a network of IBM PCs, modelled on the transputer network.

IV : A FOURTH GENERATION LANGUAGE

C.M. Iverson, *Rhodes University*

IV is a high-level language designed for use in a real-time production control environment. While most fourth generation languages are intended for use by end users, IV is more suitable for skilled professional programmers.

One of the major design objectives of IV is a dramatic improvement in programmer efficiency during application program development. The non-procedural nature of the language and the use of a number of interactive development tools provide an environment for achieving this goal.

This paper presents a language proposal for IV, and address related standardization and design issues.

QUERY DECOMPOSITION METHODS IN A DISTRIBUTED DATABASE SYSTEM

S. Lamprecht, *University of Stellenbosch*

The decomposition of queries in a distributed database system corresponds to the translation of user requests, formulated in a high level language, into sequences of instructions which retrieve data, stored in multiple tables over several sites.

The strategy used to retrieve data from the database influences the overall efficiency of the distributed data base. It is therefore very important that the decomposed query accesses the database optimally.

Several different decomposition methods exist for analysing queries for correctness and determining the best way to obtain the desired results.

The aim of this discussion is to give a brief overview of the query decomposition problem. Some of the existing decomposition methods will also be discussed.

CONSTRUCTION OF SPES INTERPRETER / COMPILER FOR C64

B. Ledwaba *University of the North*

The SPES (Specification) programming language is widely used by UNISA Computer Science students as a guide to programming and some high schools as an introduction to programming.

To my knowledge, there is no compiler for the language in existence currently. The success of this project will imply that UNISA students will not need to convert their SPES programs into other programming languages for compilation and running purposes.

The use of COMMODORE 64 was based on the cost and the availability of this type of machine in many high schools.

Since the project is still in progress, in this paper I'll address the approach I'm intending to follow ; the problems encountered so far and solutions thereof.

EXPERT SYSTEMS AS APPROPRIATE TECHNOLOGY

P. Machanick, *University of the Witwatersrand*

A definition of Appropriate Technology is supplied. This definition is used to consider how expert system technology may be applied in a Third-World context. Some problems are raised, and are used as a basis for suggesting viable approaches.

A specific approach — Knowledge Engineering Based Learning (KEBL) — is introduced as an example of an "appropriate" application of expert system technology. Experience with KEBL is detailed, including research relating to the development of software to support the KEBL idea.

Experience of using KEBL software is presented, with thoughts about further research.

COMPUTER-BASED LANGUAGE PROCESSING

M.J.D. Manamela, *University of the North*

This paper deals with the manipulation of a natural language by computer programs. The two major areas of natural language processing, namely, synthesis and recognition are outlined. Most of the details covered in this paper concern the synthesis part, which was the key theme of a project that I have done on the production of natural language speech from text in a specific language.

An outline of the appraoch and techniques for speech generation from computers is presented. Typical application areas that may incorporate natural language techniques are specified. The paper is concluded by highlighting the potential of designing and using generalised natural language processing systems in day-to-day activities.

COMMUNICATION KERNEL FOR A DISTRIBUTED DATABASE

M. Meumann, *University of Stellenbosch*

A distributed system is a system which appears to a user to be a centralised system but where multiple independent processors, performing actions on private memory and storage peripherals, are involved in carrying out actions requested by that user. This location transparency is a major goal in the implementation of a distributed database and the object of the communication kernel is to offer just that transparency.

The communication kernel is divided into two parts : the Transaction Manager and the Network Manager. Query fragments in the form of primitive operations are passed to the kernel where the Transaction Manager takes over the execution of the transaction. The Transaction Manager must ensure that the atomic requirements of a transaction are adhered to and that transactions are executed in a serializable manner. This includes handling the concurrency and integrity control for the data in the database. Management of a transaction includes passing the necessary information to the Network Manager for transmission to remote sites, and accepting messages from remote sites. The Network Manager controls access to the communication medium and handles protocols offering error free transmission.

Different methods of concurrency control have been proposed during the last few years. Our implementation will include a 'conservative timestamp' algorithm. This has been shown to be the cheapest in terms of overhead in a system which consists of homogeneous sites with horizontal control.

The talk will describe a design of the components of the kernel as well as a description of the concurrency control method. Included will be an explanation of how the kernel will be incorporated into the MINIX system and interface with the other components of the database system.

DESIGNING A STRONGLY TYPED SELF-EXTENDING DISCRETE EVENT SIMULATION LANGUAGE

H. Nell, *University of Stellenbosch*

In today's world of high technology computer systems have become crucial in the daily conduct of business. The complexity of such computer systems, communication networks and automated manufacturing is extremely high. Given the high cost of installation and upgrading, the need for powerful modelling tools to assist the system analyst is self-evident.

When choosing a simulation language a trade-off exists between the cost of learning the language and the modelling power of the language. Languages that are easy to use often set unacceptable limits on modelling power. Powerful languages are usually under-utilised as the modeller struggles with the complex constructs. Large languages often leave the impression of being ill-designed, with ad hoc extensions to overcome language deficiencies. The current acceptance level of simulation languages indicates a large conceptual distance between the respective world views of the language and the modeller, thereby discouraging use.

Wolfgang Kreutzer provides further evidence in support of a new conceptual basis :

"The extensive analysis of the various discrete event languages available today shows that all fall short of providing the powerful and flexible conceptual bases on which application oriented systems are built."

In order to achieve the required conceptual congruence, a new simulation language was designed.

ADVANCES IN SOFTWARE DEVELOPMENT AND SOFTWARE ENVIRONMENTS

M.J. Norman, *University of the Western Cape*

"If we consider software to be an information subsystem, clearly the classical approach has failed". Normally corrective and/or maintenance of software accounts for 70% of the total cost of software. In fact Boehm claims that from two to four times of the cost of development can be spent on system maintenance and Bromberg estimates that maintenance accounts for about 80% of an MIS function.

Why did the classical approach fail and how can it be improved / changed to yield information systems which will be more successful with reduced maintenance costs ?

Much of the emphasis of the classical appraoch has been on technical personnel rather than on the user. According to Aktas the most critical shortcoming of the classical approach has been the very limited reflection on user requirements in the completed information system.

According to Martin and McClure new languages should be developed for the analyst rather than for the programmer and these new languages should allow the analyst to translate his structured design, preferably built on a computer graphic screen, directly into code. This can be further extended by developing software environments that will present an end user with all the facilities or tools that will allow him to build an application from scratch to a final implemented and desirable form which will meet with all his requirements.

Past studies, according to Rushinek and Rushinek human-factors design, interfaces, user-friendliness, lower costs, application development software as a productivity aid and efficient database languages are important factors in establishing user satisfaction with a computer system.

DIE ONDERSTEUNING VAN ABSTRAKTE TOESTELLE IN PROGRAMMEERTALE

M. Olivier, *Universiteit van Pretoria*

Analoog aan die abstraheringsmeganismes wat deur programmeertale vir datatipes voorsien word, kan meganismes voorsien word om "abstrakte toestelle" te ondersteun. 'n Abstrakte toestel is 'n versameling objekte met 'n stel bewerkings daarop gedefineer; die versameling objekte kan hier ook 'n fisiese toestel (in verskillende interne toestande) wees.

Dit is eerstens nodig om aan te toon dat die voordele van abstrakte datatipes — veral die van formele spesifikasie — ook voor abstrakte toestelle geld.

Abstrakte datatipes is per definisie ook abstrakte toestelle. Die notasie wat tale soos CLU vir abstrakte datatipes voorsien kan byna net so gebruik word vir abtrakte toestelle. 'n Paar redelik eenvoudige uitbreidings is egter nodig.

As hierdie notasie by 'n Pascal-agtige taal gevoeg word, verkry die taal byna die krag van 'n vierde-generasie taal, maar is nie so toepassingspesifiek soos 'n vierde-generasie taal nie. Die koste verbonde aan die meganismes is ook nie buitensporig hoog nie.

AN OVERVIEW OF DATABASE SEMANTIC MODELLING

C. Pienaar, *University of Stellenbosch*

Database systems based on the relational model have gained wide popularity due to their ability of providing users with a simple tabular view of data and powerful relational operators for manipulating the data. By enhancing ease of use and data independence, the cost of DB-intensive application programming was reduced. In addition, the relational approach is well suited to a distributed DB environment, since it requires low communication overhead.

The main limitation of the relational model is its semantic scantiness, which hampers the expressive modelling of natural relationships and mutual constraints between entities in relational schemas. This shortcoming has motivated the introduction of various semantic data models. These models introduce high-level constructs to capture the structure of the application domain and the meaning of its data.

In selecting a data model for implementation in a distributed database environment, it seems worthwhile to try and combine the advantages of the relational approach with those of semantic models. The Entity-Relationship (ER) model, as defined by Peter Chen, models reality in terms of entities and relationships among entities, while retaining a definite relational flavour. It is also one of the most widely implemented and studied of all the semantic models. As such, it seems to be well suited for an implementation with these requirements.

VALIDATION OF PROTOCOLS WHICH ARE SPECIFIED IN A FORMAL SPECIFICATION LANGUAGE

J. Punt, *University of Cape Town*

Protocols are an important aspect of data communication. In order to ensure that a protocol is completely defined, that is without deadlock, endless looping, unspecified receptions, etc. and that the functions performed by the protocol are according to their specifications, it is necessary to validate or verify the protocol.

A common language of formal descriptions technique (FDT) is used to express protocol and service specifications in such a way that the written specifications are complete and unambiguous. One such FDT is the ISO Subgroup B's FDT known as ESTELLE (Extended State Transition Language).

In my work I will develop a validation method based on a FDT. This FDT will either be ESTELLE or a high level FDT, which will translate to ESTELLE.

I would like to exchange ideas on the validation routine, possible algorithms and their limitations (drawbacks), the FDT and how the validation routine will depend on it.

AN OVERVIEW OF A PROTOTYPE DISTRIBUTED DATABASE MANAGEMENT SYSTEM

M. Rennhackkamp, *University of Stellenbosch*

A distributed database system consists of data and a database management system which is distributed over a network of interconnected node computers. A distributed database management system can be based on one of many architectures, with a multitude of design options and numerous data models to support.

The prototype under consideration has locality, logical and physical independence as goals, together with adequate availability and efficient throughput. It is architecturally based on an integration of the ANSI/X3/SPARC database management system architecture and the ISO reference model for Open Systems Interconnection. The data is horizontally distributed over a broadcast network of homogeneous horizontally controlled nodes, with duplication according to usage. An adaptation of the Entity-Relationship model is supported, with extensions to the relationships to include update dependencies.

FORMAL SPECIFICATION OF COMMUNICATION SYSTEMS

M.M. Ross, *University of Pretoria*

Due to the complexity of communication software, it is imperative that the design of such systems should be structured. This can be facilitated by applying the concept of abstraction, which contributes to the understandability and modifiability of the software. If the system is specified by means of a Formal Description Technique such as ESTELLE, the transformation of the formal specification into a general purpose programming language results in a highly structured implementation, which promotes maintainability and reliability.

In this paper the suitability of the ESTELLE Formal Desciption Technique in the formal specification of Message Handling System protocols is investigated. A description is given of the design of a Message Handling System based on the principles of abstraction. It is shown how the features of ESTELLE can be applied to specify the Message Handling System in the most efficient way. Some consideration is given to the analyzibility of Message Handling System protocols specified according to the ESTELLE Formal Description Technique. Finally, the latest developments regarding the design of application layer protocols are discussed.

A VECTOR APPROACH TO GRAPHICAL REPRESENTATION

C.F. Scheepers, *Rand Afrikaans University*

The success of graphical representation is to a great extent dependent on the design of graphic symbols. As symbols convey concepts graphically, one should strive to represent concepts in such a way that a proper perception of the relevant concepts can be formed.

In a vector graphics environment, a limited number of basic drawing primitives serve as building blocks for the construction of graphic symbols. The visual appearance of these symbols may be changed by using primitive attributes provided by specific workstations, or even by utilizing transformation functions such as rotation, scaling and translation.

The proposed paper presents a structure and a number of routines for the construction of graphic symbols in a vector graphic environment. The results of this work can be applied in a variety of computer graphics related fields, including cartography, engineering graphics, arts, animation and solid modelling.

AN INTERFACE FOR THE MANIPULATION OF N-DIMENSIONAL DATA IN IMAGE PROCESSING SYSTEMS

J. Smit, *University of Pretoria*

Raster data used by image processing systems are stored on different devices and in different formats because of the variants in volume and application. This necessitates a data interface that is data format and device independent.

This paper discusses such an interface that extracts matrices from N-dimensional raster data. Some of the features of the interface, such as logical subimages providing a software zoom, and problems encountered in the implementation are discussed.

A PROGRAM DESIGN APPROACH AND A COMPUTER-AIDED PROGRAM DEVELOPMENT ENVIRONMENT FOR COMPUTER SCIENCE INSTRUCTION

A. Van Der Berg *University of Pretoria*

The purpose of this study is twofold: to propose a program design approach and to develop a computer-aided program development environment supporting the approach. The design approach and accompanying environment will be used for computer science instruction in the Department of Computer Science at the University of Pretoria.

Factors such as rapidly changing technology and greater exposure to computers by aspiring computer science students imply that computer science curricula and instructional methods need to be frequently revised.

A literature study indicates that there has, in general, been a shift in emphasis in the recommended contents of first programming courses. Problem solving techniques and a program development methodology are receiving increasing attention, rather than the teaching of the syntax and semantics of a particular programming language. The present study proposes a program design approach based on goals and plans as a means for problem solving. This proposal is motivated by the fact that expert programmers have been shown to use a built-in plan library from where relevant plans are extracted and coordinated according to the goals of a new problem.

The merits of computer-aided instruction and environments for program development as set out in the literature were also studied. A computer-aided program development environment can be a valuable tool for computer science instruction. The logical design for a suitable environment will be developed. Characteristics of this environment will include top-down design with stepwise refinement, syntax-directed editing, a language-independent specification method, automatic documentation, dynamic execution of plans in an experimentation mode and enforcement of good programming style. Advice-giving during every phase of program development would be a desirable feature that can be added at a later stage.

BACKCHAT TYPES

W. Van Biljon, *University of Pretoria*

backchat is a programming language designed for the construction of user interfaces. Its data type structure attempts to marry the freedom of an untyped programming language with the security of a strongly typed language. This marriage is achieved via the use of polymorphism.

Firstly the concept of polymorphism is introduced, followed by subtype polymorphism. Finally the backchat simple types, constructed types, and typing rules are presented.

DISTRIBUTED DATABASE RECOVERY METHODS: AN OVERVIEW

H. Viktor, *University of Stellenbosch*

Recovery can be defined as the act of restoring a database from an erroneous state to a usable state. In ensuring a consistent database, it is necessary to ensure that database availability is not decreased to an uneconomic, undesirable low level. This is done by using recovery control mechanisms such as log journals, archive storage checkpoints and salvation programs.

In a distributed database this process is complex as data can reside and be duplicated at multiple sites. The recovery mechanisms are much more complex, since consistency between multiple copies must be provided.

The aim of this discussion is to give an overview of the problems when assuring consistency and availability as well as highlighting the increased complexity of the situation when applied to a distributed database. A homogeneous distributed database system prototype with horizontal control distribution, providing immediate and concurrent split processing of transactions is of special interest.

A BASIS FOR APPROXIMATE REASONING

A. P. Viljoen, *University of Soth Africa*

Human reasoning is usually vague and imprecise. Approximate reasoning, based on fuzzy logic, is an attempt to formalize this. Arriving at fuzzy logic, which is based on some multivalued logic, is a two step process. The first step is to interpret predicates in the object language as fuzzy subsets of the universe of discourse. Truth-values have numerical values in the interval [0,1]. The second step is to use linguistic truth-values instead of numerical truth-values. Each of these liguistic truth-values is associated with some fuzzy subset of the interval [0,1].

SYNTHESIZING INTELLIGIBLE SPEECH FROM AFRIKAANS TEXT

M. J. Wagener, *University of Port Elizabeth*

Better quality speech can be generated from Afrikaans text if a speech synthesizer is used which is designed specifically for Afrikaans. Since such synthesizers are not available the aim of this research is to determine the characteristics of phonetic sounds in Afrikaans and to implement a software system for speech synthesis using the minimum hardware.

An introductory overview is given of an existing speech synthesis system for Afrikaans using an electronic speech synthesizer. The objectives of the research are stated and motivated followed by a brief explanation of the digitization of speech and the production of speech by the human speech apparatus. An outline is given of the features of a completed speech analysis system. In conclusion a possible model for the analysis and synthesis of speech is given.

A PRC/POSED STUDY OF REAL-TIME OPERATING SYSTEMS

G.C. Wells, *Rhodes University*

This paper discusses proposed research into operating systems suitable for real-time processing with an emphasis on process control. The methodology is to perform a detailed evaluation of the operating sytem QNX from Quantum Software Systems Ltd. The suitability of QNX for real-time programming will be investigated and compared with such systems as UNIX, MTOS-UX and OS/2. One of the important facets of the comparison is the implementation of a simulation of a large real-time process control system under each of the available operating systems. The aim is to highlight the relative strengths and weaknesses of the various systems under evaluation, and to suggest new approaches to the problems encountered in real-time programming.

STABLE C COMPILERS

G. Wheeler, *University of Cape Town*

UCT is presently engaged in the design of a RISC coprocessor for an IBM PC/AT, with an instruction set and architecture optimized for executing processes under a UNIX operating system. A large part of the project involves the UNIX port to this processor. Obviously, before UNIX can be successfully ported, a C development system for the RISC processor is required.

This talk discusses the issues involved in program portability, and examines two portable compilers with widely differing approaches : Steve Johnson's portable C compiler (PCC) and Amsterdam Compiler Kit.

REDLOG: A LANGUAGE WHICH MIXES REDUCTION AND LOGIC

D.A Wright, *Rhodes University*

Traditionally, functional languages have found their greatest use in the domain of Artificial Intelligence, where the power of features such as higher-order functions is necessary. Recently, logic languages have become more widely used by Artificial Intelligence researchers where the logical variable and non-directionality have been used to solve some problems more easily than in a functional language. These features are not available in a single language except at the cost of efficiency or an ugly interface.

A new declarative language is presented which captures the expressibility of logic languages, while gaining some of the efficiency of functional languages. This is achieved by conditional graph rewriting, with full unification as the parameter passing mechanism. The syntax and semantics are described both formally and informally, and examples are offered to support the expressibility and efficiency claims made above.

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review artilces and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Prof. G. Wiechers
INFOPLAN
Private Bag 3002
Monument Park 0105
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. Manuscripts produced using the Apple Macintosh will be welcomed. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil and the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Drawings etc., should be submitted and should include all relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:

1. Ashcroft E. and Manna Z., The Translation of 'GOTO' Programs to 'WHILE' programs., *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255, 1972.
2. Bohm C. and Jacopini G., Flow Diagrams, Turing Machines and Languages with only Two Formation Rules., *Comm. ACM,* **9**, 366-371, 1966.
3. Ginsburg S., Mathematical Theory of Context-free Languages, McGraw Hill, NewYork, 1966.

Proofs

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Only orginal papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.