J M Bishop 20/3/90

South African Computer Journal
Number 1
January 1990

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 1
Januarie 1990

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

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

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Suid-Afrikaanse Rekenaarvereniging en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083

Assistant Editor: Information Systems
Professor Peter Lay
Department of Accounting
University of Cape Town
Rondebosch 7700

Subscriptions

	Annual	Single copy
Southern Africa:	R32-00	R8-00
Elsewhere:	$32-00	$8-00

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Editorial

At last the first edition of SACJ is available. I trust that readers will find it worth the waiting. There have been a number of teething problems in getting things together, the many details of which need not be spelt out here. One significant challenge was to cope with the consequences of the resignation of Quintin Gee, QI's highly competent production editor. He assisted in the initial phases of getting this publication together but had to resign for personal reasons. It is fitting to acknowledge here not only his initial advice and assistance in getting this first issue of SACJ off the ground, but also the many hours of work that he spent in previously producing QI.

Quintin's resignation meant that a new *modus operandi* for typesetting and printing had to be established. The exercise was not only time-consuming, but also has significant cost implications. Fortunately, the Unit for Software Engineering (USE) at Pretoria University has generously agreed to sponsor this first edition. On behalf of the South African computing community, I should like to thank them for their generosity. Now that they have made a first issue of SACJ possible, it is hoped to solicit the sponsorship of one of the larger computer companies for future editions.

It might be of interest to take readers on a walk through the new journal to highlight various aspects. To begin with, the cover design follows that of several journals whose titles have the format: *The South African Journal of Subject / Die Suid-Afrikaanse tydskrif vir Vakgebied* (where *Subject* and *Vakgebied* are appropriately instantiated). While colours vary, these journals generally have *Subject* and *Vakgebied* restated on the darker portion of the cover. SACJ's title was chosen in preference to a more descriptive but also more cumbersome title such as *The South African Journal of Computer Science and Information Systems*. The appearance of the words *Computer Science and Information Systems / Rekenaarwetenskap en Inligtingstelsels* on the cover are thus out of step with the original inspiration, but seem appropriate under the circumstances.

The inside cover is of interest for several reasons. Firstly, note that Peter Lay has kindly agreed to lighten my task by acting as an assistant editor. He will deal with matters relating to Information Systems. *Contributions in this area should henceforth please be sent directly to him.* Also note that an editorial board of distinguished persons has been assembled. I should like to once again thank board members for adding status to SACJ by agreeing to serve in this capacity. They will be consulted on matters of editorial policy whenever appropriate. Finally, the subscription costs have been increased to keep pace with production costs. This increase does not affect SAICS members, who will continue to receive the journal as one of the benefits of membership.

The guest editorial by Pieter Kritzinger makes for interesting reading. Several points of concern about computer-related research in South Africa are raised. I trust that the article will focus attention on these problems and stimulate a debate which will lead to eventual solutions. It is hoped to make guest editorials a regular feature of future SACJ issues.

Of the eight research papers offered in the journal, four have been gone through the normal channel of refereeing and revisions. The remainder were submitted to the Vth SA Computer Symposium and are published here by invitation. Each paper submitted to the chairman of the symposium's program committee was sent to three referees. A ranking scheme, reflecting an aggregate measure of referee evaluation, was used as a basis for deciding on papers to be presented. After further editorial evaluation, the authors of four of the five highest ranking papers were invited to submit their papers to SACJ. While it was not possible to contact the fifth author in time for this edition, but it may be possible to publish that paper, together with a selection of others from the symposium, in future SACJ editions.

In the section marked *Communications* various items of news arriving at the editor's desk have been published. It was particularly gratifying to receive book review submissions in response to a prior general appeal. There has also been an enthusiastic response from book publishers, who have sent in a number of books for review. Titles are listed in the *Communications* section. Please contact me if you are willing to review one (or more) of these. Naturally, reviews of other books of interest in your possession will also be welcomed.

The final point to highlight in this walk through the journal is the increase in page charges indicated on the back inside cover. These reflect the increased cost of production. Since research papers in SACJ qualify for state subsidy at academic institutions, the charges should not, in general, present major problems for authors. However, it is worth pointing out that the final format of papers submitted significantly impacts on both the financial and editorial load. Submissions in camera-ready format (or nearly so) result in both a cost savings and a speed up of turn-around time by several orders of magnitude. Since many readers may not be familiar with the printing process, it may be helpful to say something about it in order to substantiate this claim.

The printing process basically involves typesetting, shooting (or photographing), and then reproduction and binding. Apart from limiting the amount of material, the printer's client has very little control over the cost of shooting, reproduction and binding. On the

other hand, anyone equipped with moderate text- or word processing facilities and a laser printer can go a long way (if not all the way) towards typesetting a paper. Even a partially typeset paper helps significantly, as I will explain below.

By typesetting I simply mean knocking the paper into the right shape and producing a laser printout. The printers regard this is a tedious, error-prone task, even if they start off with an ASCII file rather than a hardcopy of the paper. Consequently, they tend to handle large-scale typesetting by subcontracting the task. Moreover, while they may be willing to typeset uncomplicated text, they tend to balk at text containing specialized mathematical and other notation. However, they are quite skilful at cutting and pasting text, and at enlarging or reducing photographed or scanned diagrams. They are even willing to redraw sketches which are not too complicated.

As a result of the above, I have pressed several authors to do their own typesetting. In cases where it was problematic to produce double column format, a single column of appropriate width was requested. While this is a second-best option, it allows for cutting and pasting to be done by the printers. Some sketches have either been directly reduced from the author's original, while others have been redrawn by the printer. By way of exception, I have personally undertaken the typesetting of a few papers using WordPerfect. However, I would like to avoid this as far as possible in future, and consequently appeal to potential authors to make every effort to do their own typesetting.

From SACJ's point of view encouraging authors to do their own typesetting involves a compromise in that there will inevitably be slight variations in the print from one article to the next (as is in fact the case in this issue). If you are pedantically inclined, you might consider this to be a disaster. Personally, I regard it as a rather neat advertisement for the typesetting skills of SACJ contributors.

As an aside, since the handling of TEX files was initially a problem for me, I was pleased to discover that Peter Wood and his colleagues at UCT have mastered the art of producing TEX printout in the format now before you. Future authors who use TEX should consult them on details.

As to the future, it is not possible at the this stage to commit to a fixed number of SACJ issues per year. The number of issues is constrained by finance, submissions of the right quality, and time available to the editorial staff (including our anonymous and unsung heroes - the referees). The ideal is to produce four issues per year, but this may not always be attainable.

In conclusion, if readers have as much fun in reading this first issue of SACJ as I have had in editing it, the hours spent on it will have been well worthwhile. Hopefully SACJ is destined not only to be a permanent feature of the Southern African computing scene, but also to significantly contribute to research in the region.

Derrick Kourie
Editor

This SACJ issue is sponsored by
The Unit for Software Engineering
(USE)
Department of Computer Science
Pretoria University

Guest Editorial

Funding Computer Science Research in South Africa

P S Kritzinger
Department of Computer Science, University of Cape Town, Rondebosch 7700

The word *research* has many connotations and is often abused. In everyday language a person does not simply *search for information in a library*, for example, but rather does *research*, thus pretentiously conferring an aura of intellectual activity on an effort which requires very little original thought.

Here I will interpret the term to mean work which generates results that gain international recognition. This implies that the work is published in good international journals or presented at international conferences. I believe this is the only valid index of the quality of research.

With very few exceptions, the computer industry in South Africa is a consumer of computer technology, rather than a developer. In contrast with, say, the chemical industry, there is therefore no tradition of research in computer science in the South African computer industry and computer science researchers therefore have, as virtually their only source of funding, the Foundation for Research Development (FRD) which has its origins in the CSIR.

The FRD was formed in April 1984 with the development and use of research expertise in the natural and applied sciences and engineering as its mission. This mission is primarily directed at the universities, museums and technicons with the ultimate aim of improving the life of all South Africans.

Although the FRD has several programmes, the two which are of main concern to computer scientists are the Core Programmes and the Special Programmes.

FRD Core Programmes foster the optimum development of a scientific and technological knowledge base by supporting individual self-initiated research. These programmes, started only about 4 years ago, have met with considerable acclaim, particularly in regard to the way in which research funding for a particular individual is decided. To qualify for support within a Core Programme, researchers must obtain a certain evaluation status within the FRD and funding is then linked directly and exponentially to the merit of the individual concerned, rather than being linked to the specific project proposed.

In the evaluation process, peer review is strongly emphasised. The researcher himself is expected to nominate referees, whose status and reports play a decisive role in the evaluation. As a result of this evaluation, an applicant is assigned a specific evaluation status category. There are currently 9 categories in all, but the ones of main interest are:

- **A** researchers who are without any doubt accepted by the international community as being amongst the leaders in their field (52);
- **B** researchers not in category **A** but who nevertheless enjoy considerable international recognition as independent researchers of high quality (182);
- **C** proven researchers who have maintained a constant high level of research productivity and whose work is regularly made known internationally, or proven researchers whose current research output is less but who are actively engaged in scholastic activity (433);
- **P** researchers younger than 35 years of age who have already obtained a doctoral degree and who have shown exceptional potential as researchers (10); and
- **Y** young researchers usually under 35 years of age, who are highly likely to achieve C status by the end of their support period (108).

The number of researchers in the various categories as of August 1989 has been indicated in parentheses above. Of these, only 7 persons are computer scientists: 1 in category B; 3 in category C; and 3 in category Y. Only 4 departments of computer science are involved.

The other main programmes of concern to computer persons are the Special Programmes which aim at developing research manpower in priority areas. After identification of an area that merits particular research development, given local expertise, a Special Programme is launched to address the problem in the national interest.

Although a manager of a Special Programme has to be an FRD evaluated researcher, the same need not be true for the other team members. Regular peer evaluation of researchers as well as evaluation of the progress and results of Special Programmes are considered essential. Special Programme awards will be made for the first time towards the end of 1989. It is therefore not yet known whether proposals already submitted for programmes in computer science have been successful.

It is clear that, in the context explained above, there is virtually no computer science research being done in South Africa - a scary thought which has considerable implications for this country! Why is this so? There are several reasons, but I would like to single out two in particular.

Qualified faculty and students is an abiding problem at the heart of computer science departments. Acquisition of new faculty members is an issue intimately linked to the number of graduate students successfully completing PhD degrees. This problem is by no means unique to South Africa. For instance, data gathered in North America indicates that in 1983 there were over 200 vacancies in the 91 departments that have doctoral programmes in computer science. At the same time, only approximately 250 PhD's were granted in North America - a figure that has remained relatively unchanged for the past several years. A large number of those graduates were attracted to industry and industrial research laboratories. Although I do not have solid data at my disposal, I would think that South Africa produces at most one PhD graduate in computer science per year. There are currently 20 departments of computer science at universities is South Africa. It will therefore take us 20 years to locally produce one new faculty member with a PhD in computer science for every university.

Contributing to the above problem is our current academic image. The graduate student usually sees concerned computer science faculty members as rather harried individuals, having large undergraduate classes, much committee and professional work, and labouring under an ill-fitting model (applicable to more established disciplines) for decisions on tenure, salary and promotion. Further, as undergraduates, many prospective graduate students were not engaged in research projects involving computer science faculty, and for that reason were not exposed to graduate students doing research, and rarely developed a camaraderie with any computer science professionals. At last count there were only 5 individuals in South Africa who completed their computer science doctorate at a university outside South Africa where they had the good fortune to work in an environment in which sufficient faculty and funds were available to create an ethos of research. It is difficult to convince students that their interests and goals can be served by a PhD in computer science or by an academic career.

The second problem, which is of greater concern to me since there is no immediate solution to it, has to do with the fact that senior persons who decide the fate and fortune of academic computer science departments are, in general, individuals whose professional careers started well before computing machines came into every day use - that is to say, in the years B.C. (Before Computers). These persons of influence do not always understand what "computers" are, and what their potential influence upon the workplace in particular and society in general are. As far as research (as opposed to teaching) is concerned, most of them understand that a medical school needs special and expensive equipment (not to mention, expensive faculty) and that engineers must have a workshop and special machinery to teach their students and conduct research. They understand that if one needs to build up a defense industry, it will cost billions of rands; but they are not so sure about computer science, even though many other countries have recognised it as of national strategic importance.

I believe that only time and dedication will lead to a solution of these seemingly insurmountable problems and allow computer scientists to take their rightful place in the research community in South Africa.

Bibliography

P J Denning, [1981], Eating our seed corn, *Communications of the A.C.M.*, **24**(6), pp.341 - 343.

J Tartar (Ed.), [1985], The 1984 Snowbird Report: Future issues in computer science, *Communications of the ACM*, **28**(5), pp490 - 493.

D Gries, R Miller, R Ritchie and P Young, [1986], Imbalance between growth and funding in academic computing science: two trends colliding, *Communications of the A.C.M.*, **29**(9), pp.870 - 878.

J E Hopcroft and D B Krafft, [1987], Towards better computer science, *IEEE Spectrum*, pp.58 - 60.

A Design Environment for Semantic Data Models

J Kambanis
Department of Computer Science, UNISA, Pretoria.

Abstract

This paper describes the functionality and architecture of a Semantic Data Modeling environment. It integrates User Interface techniques used in Computer Aided Software Engineering and Expert Systems shells into a system of interacting views of the Semantic Data Model. Each view provides a different way of viewing and modifying the model. Hypertext is an integral part of the environment. The architecture of the environment is layered, consisting of an associative network layer, a semantic functions layer and a user interface, and is aimed particularly at maximizing its generality and extensibility.
Keywords: *Semantic data models, Software engineering environments, data bases, hypertext.*
CR categories: *H.1*

Presented at V th S.A. Computer Symposium.

1. Introduction

Some integration of ideas from Knowledge Representation and Data Bases has created the area of Semantic Data Models which has attracted interest in recent years [15]. Semantic Data Models provide a conceptual schema which is at a higher level than the computer-oriented and record-oriented models, and are particularly suited for designing data intensive applications. Languages like Taxis [20] and Galileo [1,2] are based on a Semantic Data Model, and can be used to model Information Systems.

The objective of this work was to develop an environment which uses advanced User Interface techniques to support the design of Semantic Data Models. The environment should also be (architecturally) easy to change in order to facilitate *its* evolutionary development. In this first prototype, the design environment is based on the Taxis Semantic Data Model.

2. Summary of Taxis Semantic Data Model and Language

This section contains a summary from [21] which will serve as a very brief introduction to Taxis.

The Taxis representational framework consists of two main ingredients: an object oriented framework and a set of abstraction principles. Taxis emphasizes object organization along different abstraction dimensions over message passing. An abstraction mechanism is a conceptual or linguistic mechanism that allows certain information to be highlighted while suppressing other information. The Taxis framework offers a set of complementary abstraction facilities based on the notions of aggregation, classification, and generalization.

- *Aggregation:* If we define an *attribute* to be a directed relationship between two objects, aggregation allows one to view an object as a composite of the objects to which it is related by attributes. For example a person has a name , an address, and so on.
- *Classification:* Along an orthogonal dimension, the classification abstraction allows individuals to be grouped into classes that share common attributes. A class represents a generic concept, such as "person".
- *Generalization* (and its converse, *specialization*) allow the common attributes of several classes to be abstracted into the definition of a single, more general, class. For example, the class of persons can be represented as a generalization of classes representing males, females, managers, engineers, etc.

The Taxis conceptual framework for modeling the world provides objects (or *entities*) as the basic building block for constructing descriptions. Objects which are individuals are called *tokens*, and objects which are collections of tokens are *classes*.

Objects can be related to each other through *factual attributes*. A factual attribute has three components: a *subject*, an *attribute* and a *value*, and can be represented by a triple such as:
<johnSmith, age, 26>.

Generic information that pertains to all instances of a *class*, is represented through *structural attributes* which can be represented by a triple such as [Person, age, AgeValue]. This triple is the general statement that every instance of the class person has an age factual attribute whose value comes from the class AgeValue. Similarly triples can be used to represent specialization, such as: (Patient, isA, Person).

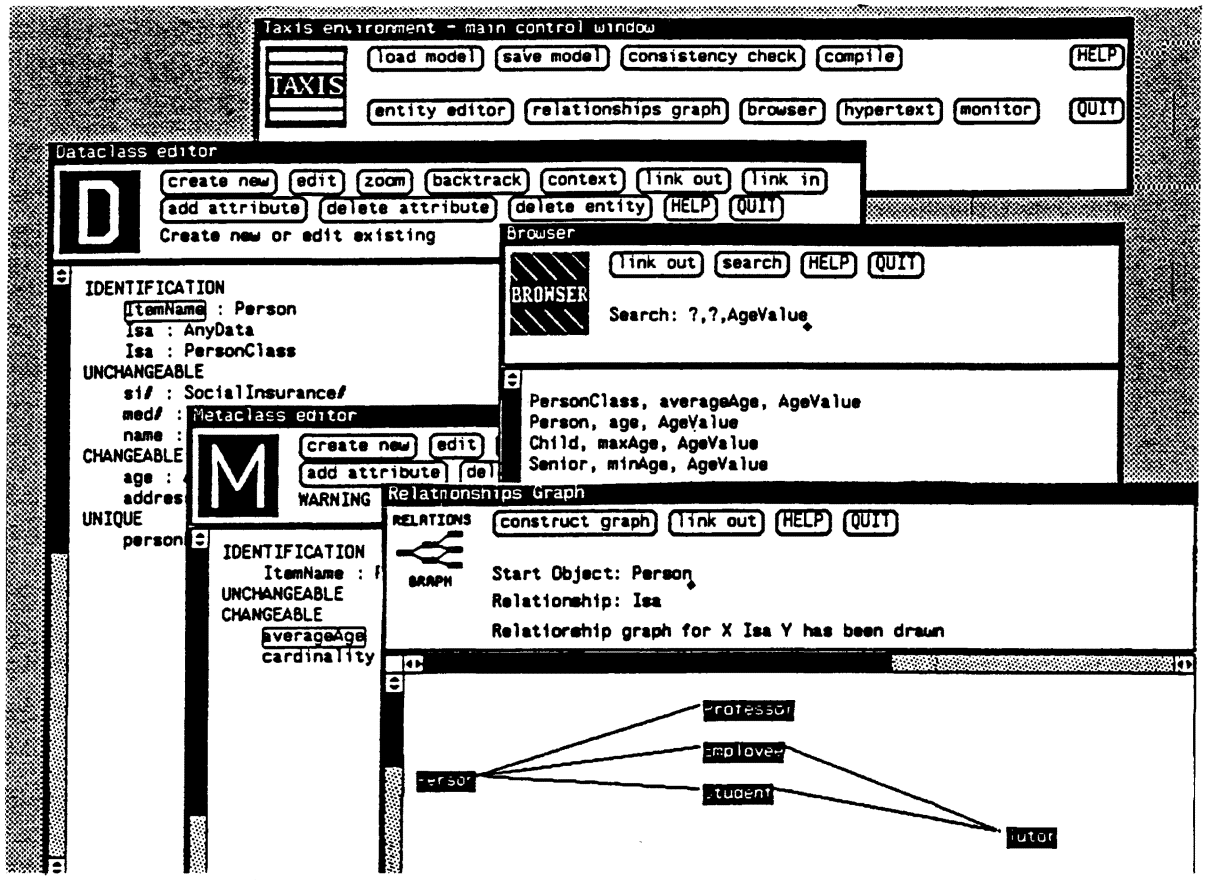

Figure 1: Example of a Taxis Design Environment Screen

The Taxis language is used to define the attributes of entities such as *data classes* and the relationships between such classes.

Transaction classes are used to define operations that can be performed on data classes. For example the AdmitPatient takes a person and makes it a patient after checking certain conditions.

Taxis also provides *script classes* as a modeling tool for long term processes. For example pacemaker-Patient is a script that describes the process of dealing with a pacemaker patient. For every pacemaker patient there is an instance of this script, which remains active from the time his doctor decides to have a pacemaker implanted until the time he stops the treatment. Such a script is represented by a modified petri net which consists of *states* and *transitions*.

More details on Taxis are in [19,20,29,22,8 and 3].

3. Overall Functionality of Taxis Design Environment (TDE)

Good representational frameworks and methodologies for information systems development need to be supported by an effective design environment [6]. Sophisticated techniques for man-machine interfaces [26] have been applied to Computer Aided Software Engineering (CASE) and Artificial Intelligence [11], [4]. TDE integrates tools and techniques that have been shown to be useful in practice.

A designer needs to look at the target model from different angles and at different levels of detail. In TDE the approach is to maintain an internal representation of all the knowledge regarding the object being described, and offer the designer different *views* for inspecting or manipulating the model. Each view presents a specific *extract* of knowledge from the total model in a form most suitable to support one of the designer's *mental concepts*, each concept representing a useful way of looking at the model.

Figure 1 shows a typical screen of (TDE) screen showing the system of views, in this case consisting the Main-Control view, two Class Editor views, the Relationship Graph view, and the Browser view.

More details about each view are given in the following sections.

4. The Entity Editor Views

A basic facility that the designer of a Semantic Data Model needs, is to create and define entities. Entity Editor views are those views which offer the design-

er a window with a structured editor for editing *specific* entities. The behavior of each structured editor depends on the type of entity being edited, and it accepts only specific user actions which are relevant for modifying the structure of that type of entity.

The Taxis entities handled by the TDE are: data class, metaclass, integer class, enumeration class, transaction class, script, and token. Figure 1 shows an example of a Data Class entity. This shows the structure of the generic entity **Patient** which inherits through an **isa** relationship the structural attributes of Person. Using this view the designer modifies a Data Class entity by adding or deleting attribute/value pairs using keyboard or mouse control.

An important component of all views is the *focus*. In figure 1 this is the square rectangle which at any time encloses one of the constituents inside the view. For example in figure 1 the focus is on the attribute value Person. Different actions can be taken relating to the item in focus (e.g. the attribute can be deleted).

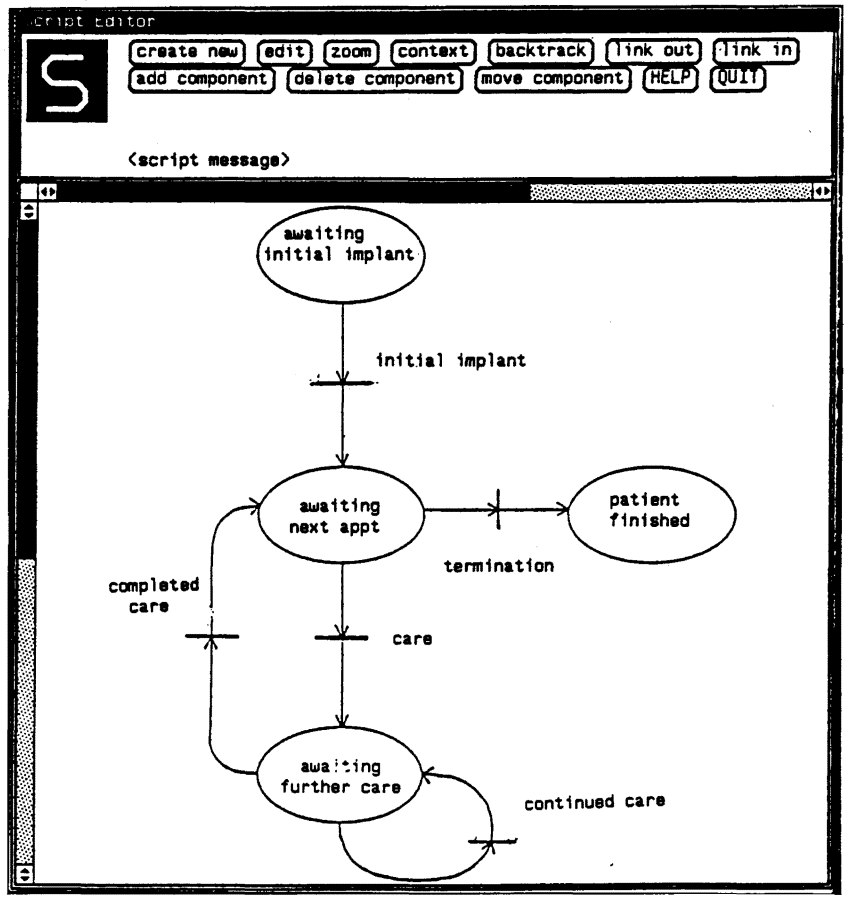

Figure 2: A Script View

When working with a Semantic Data Model, the designer needs (among others things) the facility to instantly view or edit entities which are directly *related* to the current entity. TDE provides this facility from any Entity Editor. For example the entity Patient (figure 1) is related to entities Person, HospitalWard, etc. By putting the focus on Person and selecting **"zoom"** the entity Person can be brought into the editor. By repeated use of this, the designer can *navigate* through the knowledge base. For example he can view all inherited attributes of a certain class, by leapfrogging through the sequence of **isa** relationships. Backward navigation can also be done. By selecting **"context"** the designer can view any of the entities that *refer to* the entity currently in the editor. He can also (chronologically) backtrack to the entity previously in the editor, by selecting **"backtrack"**.

The Entity Editor may be graphical. A Taxis Script is a long term process involving states and transitions, represented by a petri-net type of diagram. The Script View (figure 2) offers the facility for displaying and editing script diagrams.

Although different modes of presentation are used to suit the type of entities being edited, the same mechanisms of focus and navigation apply in all entity editors.

Instances of generic objects are Tokens. The Token Editor view is an entity editor similar to the class editors but with significant differences. On creating a new token, TDE offers a menu of relevant classes. On selecting a specific class, a Token is created which contains *all* attributes inherited from the selected class and its superclasses, with empty slots to receive attribute values from the designer. As the focus moves onto a certain slot, information regarding the *type* of that value (as specified by the inherited structural attributes) appears in the control panel of the Token view. This is the constraint against which the value in focus will be validated.

5. The Relationship Graph View

The Entity editors, as described above, enable the designer to create and edit *single* objects. The Relationship Graph view (figure 1 and 5) gives the designer the "big picture" by presenting a number of entities and the relationships between them. An overall view showing all relationships would confuse the designer by providing too much information.

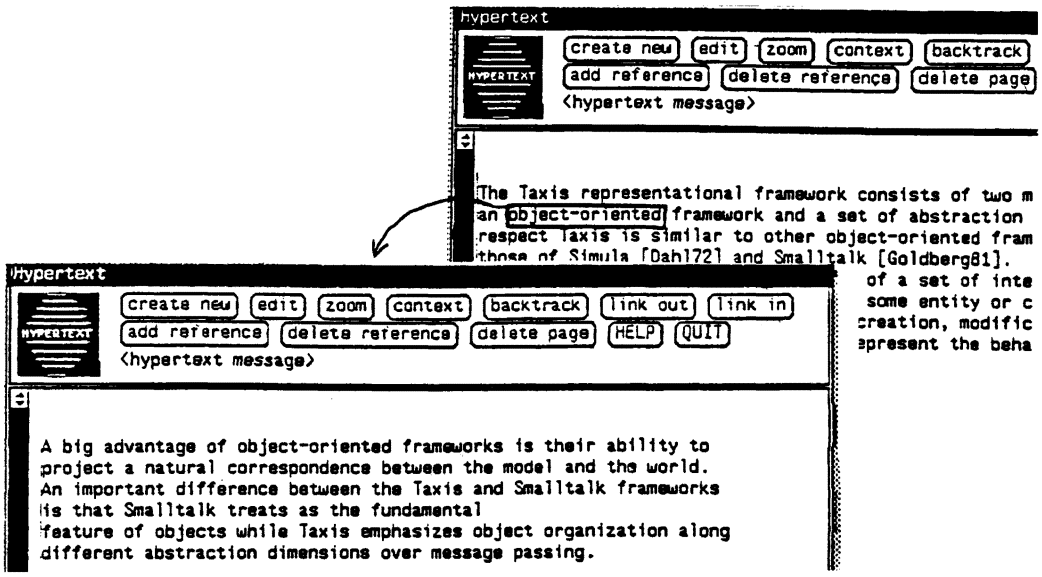

Figure 3: A Hypertext View

For this reason the Relationship Graph view provides *filtering* as required by the designer. The control panel (figure 1 and 5) contains a slot for the name of the relationship, which is to be shown in the graph, at the exclusion of others. Also by filling the **"top object"** slot the entity at the top of the hierarchy is selected, thus restricting the graph further.

Here, as in all other views, the focus can be moved to a specific entity shown and by selecting the **Zoom** command, the details of that entity can be shown in an Entity Editor view.

As the designer changes the knowledge base through Entity Editor views, the Relationship Graph View gives him an updated overview of the model.

6. The Browser View

The designer of a Semantic Data Model, should also able to search for any entities that satisfy given conditions. This first prototype of TDE offers a simple query facility, which supports the type of queries envisaged at this stage. A query consists of an object/attribute/value triple, and returns all entities that match this template. For example, figure 1 shows all entities and attributes with value AgeValue.

7. The Hypertext View

Almost in all activities within software engineering and modeling in general, there is a need for "freehand" textual descriptions. Hypertext [9] is used here to enhance Semantic Data Model design. In TDE Hypertext is integrated naturally as another type of entity or concept, which can be edited through a Hypertext view. As in the other entity editors the focus can be moved to any reference in the text and a **"zoom"** done through that relationship to another block of Hypertext. Similarly backward navigation is possible through **"context"** (figure 3).

Hypertext objects can be related (linked) to other entities. For example, by focusing on any attribute in an Entity Editor view, a Hypertext view can display and edit Hypertext associated with that particular attribute (figure 4).

In this way comments associated with the entity can be as extensive as desired, without confusing the formal entity description, and each comment may itself be organized in layers of detail. Furthermore a comment can be associated with more than one entity or specific attributes.

8. Inter-view Communication and Linkage

The four basic views described above (the Entity Editors, the Relationship Graph, the Browser, and the Hypertext), each offers a specific type of facility for viewing and changing the model. This has been further enhanced by *linking* views into coordinated pairs.

For example, a Relationship Graph can be *linked* to an Entity Editor view so that while the *focus* in the Relationship Graph is moved from one entity to the other, the Entity Editor view *dynamically* shows and edits the entity currently in focus (figure 5).

9. Consistency and Completeness

Since a Semantic Data Model is normally created and shaped *gradually*, it is necessarily in an inconsistent and incomplete state during its development.

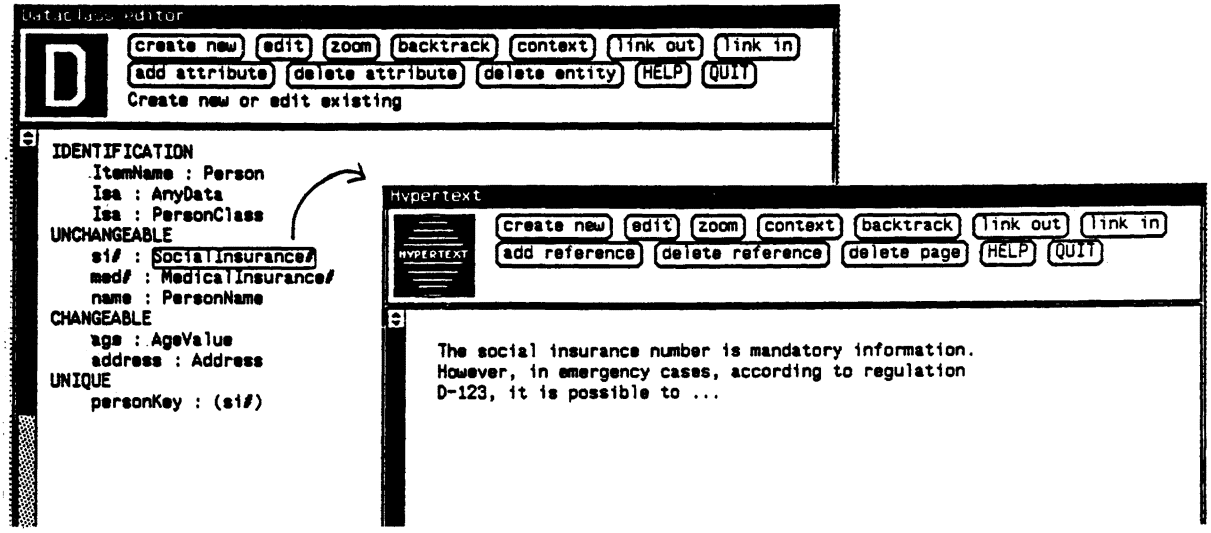

Figure 4: An Attribute Associated with Hypertext.

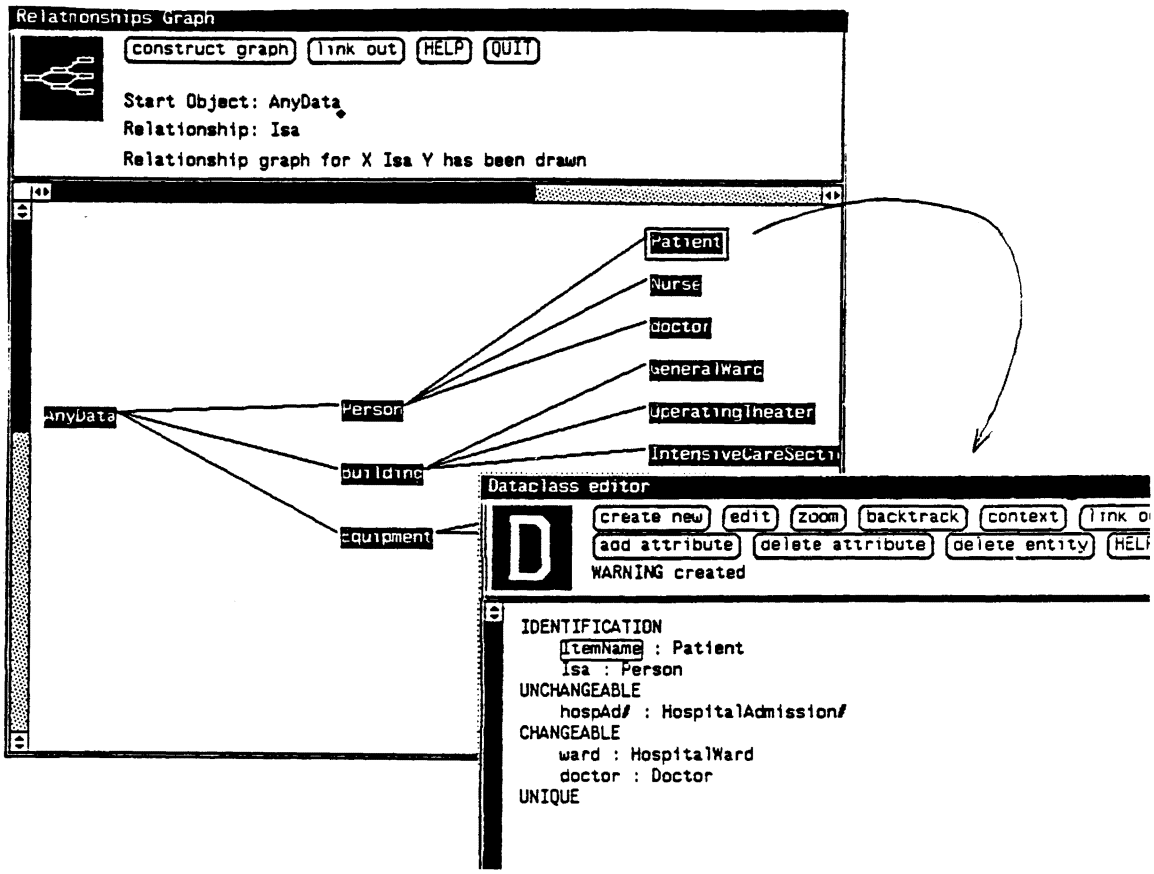

Figure 5: A Relationship Graph Linked with an Entity Editor

TDE takes actions and responds with error messages in real-time where appropriate, while the overall consistency and completeness checking for the whole model, is done on demand by generating a Taxis program that represents the model and passing it to the compiler. The environment can then display the error report produced by the compiler.

10. Architecture of the Taxis Environment

For the sake of extensibility, TDE has a layered architecture consisting of:

- User interface;
- Semantic functions (Taxis concepts); and
- Associative network.

The bottom layer is an *associative network*, containing no semantics, merely labeled associations between concepts (nodes). This associative network offers a set of primitives which may be used to manipulate this network. These primitives can be used to create nodes, to link nodes with labeled arcs, to navigate through selected arcs in the network, etc.

On top of this associative network there is a set of functions which implement the *semantics* of the *particular* knowledge representations schema (in this case Taxis entities). These functions make use of the lower level, associative network primitives.

This implementation of the knowledge base has the advantage that when another KR scheme needs to be implemented (or the present one extended) we only need to create the additional set of semantic functions.

Finally a user interface layer resides at the top, consisting of all the different views and their editors. More views can be added to this layer.

11. Related Work

When comparing this environment with a previous one for Taxis (Taxied) [24], the multiview capability and graphical interface of this prototype is an improvement over the command and text oriented interface of Taxied.

A survey of some experimental environments based on Semantic Data Models is given in [15]. When compared with these, the overall feature of the environment presented here is that it integrates into a coherent system, functionalities used individually by these environments.

For example, LID [13] is an environment oriented towards browsing by traversing the schema from one object to related objects, with no capability to view sets of objects. Others, like GUIDE [28], support selection-like queries, while DDEW [Hull 87] represents the schema using graph structures. The Taxis Design Environment provides all these facilities in integrated form.

Some facilities are handled differently. For example ISIS [14] and SNAP [15] allow the designer to create and manipulate the visual representation of the schema, whereas the Taxis Design Environment does not. Instead it uses an Entity Editor view to create and edit entities and their relationships, while at the same time the Relationship Graph view shows the updated graphical representation of the model, which it produces automatically from the model using a predetermined format.

This prototype of the Taxis Design Environment shares a weakness with all these environments, except DDEW and SKI [15]: it does not yet interface with a disk-based database. However an independent prototype implementation of the Associative Network infrastructure has been created on top of a relational database and needs to be integrated into TDE (using the Empress database).

A feature of this environment is the natural integration of Hypertext with the other functionalities of Semantic Data Modeling environments. This enables the designer to incorporate in the model, informal descriptions at different levels of detail, linked to different entities.

The main distinguishing feature of this environment is the use of a system of multiple, independent views of the Semantic Data Model and the ability to link these into coordinated pairs.

More information on the Taxis Design Environment is available in [17].

12. Implementation

This TDE prototype as a whole is in working order with several of the views functioning as described (data class editor, transaction editor, relationships graph, hypertext editor) while others are being developed as per the functional design described above. This prototype has been developed in C under UNIX on a Sun workstation using Sunview. An object-oriented design framework has been used. Each view is implemented as an object, consisting of its internal state variables and a set of procedures. More than one instances of each view can be created. A form of message passing is used for communication between views.

13. Future Work

The Taxis Design Environment is now an ongoing project at the University of Toronto. Present work is concentrated in creating additional views and a tighter integration between TDE and a Taxis compiler. The long term aim is to use the generality and extensibility of TDE to evolve towards a Global Information Systems modeling environment [5] which would provide integrated, knowledge based support for the entire system development life cycle.

Acknowledgments

This work was performed at the University of Toronto while on sabbatical from the University of South Africa. I wish to thank Professor John Mylopoulos for his support and guidance throughout this project, as well as Brian Nixon, David Lauzon and Lawrence Chung for their suggestions.

Bibliography

[1] A Albano, L Cardelli and R Orsini, [1985], Galileo: A Strongly Typed, Interactive Conceptual

Language, *ACM TODS*, **10**(2).
[2] A Albano, [1985], Conceptual Languages: A Comparison of ADAPLEX, Galileo and Taxis, *Proceedings of the Workshop on Knowledge Base Management Systems*, Crete, 343-356.
[3] J Barron, [1982], Dialogue and Process Design for Interactive Information Systems Using Taxis, *SIGOA Newsletter*, **3**(1,2), 12-20.
[4] S Becker and B Selman, [1986], Overview of Knowledge Acquisition Methods for Expert Systems, *Technical Report CSRI-184*, University of Toronto.
[5] A Borgida, M Jarke, J Mylopoulos, W Schmidt, and Y Vassiliou, [1987], The Software Development Environment as a Knowledge Base Management System, *Technical Note CSRI-46*, Computer Systems Research Institute, University of Toronto.
[6] J T Brady, [1986], A Theory of Productivity in the Creative Process, *IEEE CGA*.
[7] P S Chen, [1976], The Entity-Relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems*, **1**(1), 9-36.
[8] K L Chung, D Rios-Zertuche, B A Nixon, and J Mylopoulos, [1988], Process Management and Assertion Enforcement for a Semantic Data Model, In J W Schmidt, S Ceri and M Missikof (eds), *International Conference on Extending Database Technology, Venice, Italy, March 1988, Proceedings*, Lecture Notes in Computer Science, No 303, Berlin: Springer-Verlag, 469-487.
[9] J Conklin, [1987], *Hypertext: An Introduction and Survey, IEEE Computer.*
[10] O J Dahl, and C A R Hoare, [1972] Hierarchical Program Structures, in O J Dahl, E W Dijkstra, and C A R Hoare, (eds.), *Structured Programming*, Academic Press.
[11]R Fikes, and T Kehler, [1985], The role of Frame-Based Representation in Reasoning, *Communications of the CACM*, **28**(9).
[12]N Findler, (editor), [1979], *Associative Networks*, Academic Press.
[13]D Fogg, [1984], Lesson from "Living in a database" graphical query interface, *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Boston, Mass.), ACM, New York.
[14]K J Goldman, S A Goldman, P C Kanellakis, and S B Zdonik, [1985], Interface for a semantic information system, *Proceedings of the ACM SIGMOD International Conference on Management of Data* ACM, New York.
[15]R Hull, and R King, [1987], Semantic Database Modeling: Survey, Applications, and Research Issues, *ACM Computing Surveys*, **19**(3).
[16]M Jarke, M Jeusfeld, and T Rose, [1987], *A Global KBMS for Database Software Evolution: Design and Development Strategy*, Universität Passau, Germany.
[17] J Kambanis, [1988], An Information Systems Design Environment Based on a Semantic Data Model, *CSRI Technical Report CSRI-210*, University of Toronto.
[18] M Koubarakis, [1988], Telos: A Knowledge Representation Language for Requirements Modelling, *Technical Report draft*, Dept of Computer Science, University of Toronto.
[19] J Mylopoulos, M P A Bernstein, and H K T Wong, [1980], A Language Facility for Designing Interactive Database-Intensive Applications, *ACM Transactions on Database Systems*, **5**(2), 185-207.
[20] J Mylopoulos, and H K T Wong, [1980], Some Features of the Taxis Model, *Sixth International Conference on Very Large Data Bases*, 399-410.
[21] J Mylopoulos, A Borgida, S Greenspan, C Meghini and B Nixon, [1986], Knowledge Representation in the Software Development Process: A Case Study, In H Winter (ed), *Artificial Intelligence and Man-Machine Systems*, Lecture Notes in Control and Information Sciences, No 80. Berlin: Springer- Verlag, 23-44.
[22] B A Nixon, [1983], A Taxis Compiler, *M.Sc. thesis*, Dept of Computer Science, University of Toronto.
[23] B A Nixon, L Chung, D Lauzon, A Borgida, J Mylopoulos and M Stanley, [1987], Implementation of a Compiler for a Semantic Data Model: Experiences with Taxis, In U Dayal and I Traiger (eds), *ACM SIGMOD '87, Proceedings of Association for Computing Machinery Special Interest Group on Management of Data, 1987 Annual Conference,* San Francisco, CA, *SIGMOD Record*, **16**(3) 118-131.
[24] P D O'Brien, [1982], Taxied: An Integrated Interactive Design Environment for Taxis, *M.Sc. thesis*, Dept of Computer Science, University of Toronto.
[25] M R Quillian, [1968], Semantic Memory, in M Minsky (ed), *Semantic Information Processing*, MIT Press, 227- 270.
[26] B Shneiderman, [1987], *Designing the User Interface*, Addison-Wesley.
[27] J Smith and D C P Smith, [1977], Database Abstractions: Aggregation and Generalization, *ACM Transactions on Database Systems*, **2**(2), 105-133.
[28] H K T Wong, and I Kou, [1982], GUIDE: A graphical user interface for database exploration, *Proceedings of 8th International Conference on Very Large Data Bases*, Very Large Data Base Endowment, Saratoga, Calif.
[29] H K T Wong, [1983], Design and Verification of Interactive Information Systems Using TAXIS, *PhD Dissertation*, University of Toronto.

Computers and the Law

Submitted by Antony Cooper
CSIR

The SA Law Commission has established a commission on "The Legal Protection of Information".

The commission is still in its preliminary stages and the assigned researcher, Mr Herman Smuts, is still preparing the working paper. He does not know when it will be finished, but once the working paper has been prepared, they will invite comments for about two years, before preparing the final report. I have contacted Mr Smuts, and he would be most grateful to receive input at this stage, especially regarding the terms of reference of the commission. His address is:

C/o SA Law Commission
Private Bag X668
PRETORIA
0001

In addition, there is an ad-hoc committee at the Registrar of Copyright investigating numerous copyright issues, including those relating to software and data. Mr Smuts' commission will be liaising with the ad-hoc committee.

I feel that SAICS has an obligation to submit evidence to the commission, and I would appreciate it if you would circulate the members of the Council of SAICS, and perhaps the general membership as well, to solicit ideas concerning SAICS's input.

I shall prepare something for the commission, either in my personal capacity, or in my professional capacity here at CSIR. I would be willing to assist in the preparation of any evidence SAICS might submit.

4th National MSc/Phd Computer Science Conference

Report by Danie Behr
University of Pretoria

This conference was held from 7th to 10th September 1989 at the Cathedral Peak Hotel in the Drakensberg. The conference was attended by 61 postgraduate students from 11 South African universities. Most were engaged in MSc studies, although 5 Phd students also attended. These numbers are encouraging for the South African computer science community. This type of conference is rather unique in that it affords students the opportunity of sharing their research, and getting to know other researchers in the country. The number of Afrikaans and English speaking students attending the conference were roughly equal. Presentations were made in the language preferred by the student. Invitations were sent to all universities with computer science departments. The conference was organized by the students themselves.

Some of the more popular research topics that were presented included expert systems, data communications, computer security, graphics, software engineering, user interfaces and data bases. The main sponsor for this year's conference was the Division for Microelectronic Systems and Communication Technology of the CSIR. The conference was opened with an interesting talk on the myths and motivations of post graduate studies by Prof DG Kourie, acting head of the Computer Science Department at Pretoria University.

The next conference will be presented by the University of Port Elizabeth. People requiring further information about the next conference should contact Andre Calitz, Charmaine du Plessis or Jean Greyling of the Department of Computer Science at UPE.

A list of authors and papers presented at the symposium follows:

S Crosby, University of Stellenbosch
Performance Analysis of Wide Area Computer Communication Networks

A B Joubert, PU for CHE Vaal Triangle Campus
Image Processing Libraries

A Calitz, University of Port Elizabeth
An Expert System Toolbox to assist in the classification of objects

L von Backström, University of Pretoria
Integrated Network Management

R Foss, Rhodes University
The Rhodes Computer Music Network

A McGee, University of Natal
On Fixpoints and Nondeterminism in the Sigma-Lambda Calculus

P G Mulder, Randse Afrikaanse Universiteit
A Formal Language and Automota approach to Data Communications

A Tew, Randse Afrikaanse Universiteit
Drie dimensionele grafiek grammatikas

T C Parker-Nance, University of Port Elizabeth
Human-Computer Interaction: What Determines Computer Acceptance

E Coetzee, PU vir CHO Vaaldriehoekkampus
Opsporing van rande in syferbeelde dmv verskerping en drempelbepaling
D A Sewry, Rhodes University
Visual Programming
A Cooper, University of Pretoria
Improvements to the National Exchange Standard
E S Badier, University of Port Elizabeth
A Computer Assisted Diagnostic System (CADS)
C du Plessis, Universiteit van Port Elizabeth
Persoonsidentifikasie dmv naampassing in 'n genealogiese databasis
J Greeff, University of Stellenbosch
The Entity-Relationship Model and its Implementation
D A de Waal, PU vir CHO
Flat Concurrent Prolog (FCP) en Flat Guarded Horn Clauses (FGHC): 'n Vergelyking
E Naude, UNISA
Interne metodes in Liniêre Programmering
A Deacon, University of Stellenbosch
Global consistency in non-locking DDBMS
A Wilks, Rhodes University
The Synchronisation and Remote Configuration of the Resources in a Computer Music Network
J Greyling, University of Port Elizabeth
The design of a User Interface with special reference to an Interactive Molecular Modelling Program
L Drevin, PU vir CHO
Rekenaarsekuriteit: Verskillende vlakke van kontrole
Dieter C Barnard, University of Stellenbosch
The design and implementation of a modest, interactive proof checker
R A Schmidt, University of Cape Town
Knowledge Representation Systems and the Algebra of Relations
J Hartman, Randse Afrikaanse Universiteit
Die Gebruik van Objek-georiënteerde Programmering in die Moderne Sneltrein Omgewing
S Lawrie, Rhodes University
The Design and Implementation of a System for the Interactive Control of a MIDI-based Studio
E Mulder, Rand Afrikaans University
A Formalisation of Object-Oriented Principles
C J Tolmie, UOVS
Die Ontwikkeling van 'n Ekspertrekenaarstelsel vir die beoordeling van die resultate van die Technicon H1-Bloedselanaliseerder
R Breedt, University of Pretoria
Realism with Ray Tracing
J van Jaarsveld, University of Pretoria
Developing Medical Expert Systems: A knowledge acquisition perspective
W Appel, University of Pretoria
TCP/IP Implementation on Ethernet
E Goedeke, University of Natal
Eggspert's Control Structure
M Harmse, University of Stellenbosch
Modelling of I/O Subsystems
H L Viktor, University of Stellenbosch
A Quantitative Model for Comparing Recovery Techniques in a Distributed Database
M Olivier, Randse Afrikaanse Universiteit
Rekenaarvirusse in Suid-Afrika

Book Reviews

An Introduction to Functional Programming Through Lambda Calculus
by Greg Michaelson, Addison-Wesley, 1988.
Reviewer: Dr. E P Wentworth, Rhodes University

Recently we have seen a number of excellent *second generation* texts on Functional Programming. Michaelson's text assumes some previous programming experience with imperative languages, and presents the functional approach as an alternative paradigm. He begins with a very accessible exposition of the Lambda Calculus, and carefully develops this foundation to encompass the important aspects and paradigms of functional programming. The programming notation is language-independent, although the last chapters are devoted to a brief look at two specific languages, Standard ML and Lisp. The examples and exercises are mainly utility in nature, e.g. "insert a sublist after the first occurrence of another sublist in a list", and can generally be solved in a couple of lines. Answers to the exercises are provided in an appendix.

The approach is slanted towards developing a solid base for understanding functional languages and computing. In this respect the book achieves a good balance between the theoretical underpinnings and their practical application. On the practical side, however, I found the lack of more substantial examples and exercises disappointing. Most programming texts tackle a set of 'standard' problems which are well-understood in the academic community and provide an informal benchmark for comparisons. Since the book is targeted for those already versed in imperative languages and standard algorithms, one might expect the examples to clearly demonstrate the elegance and power of the *problem-oriented* functional approach in these areas. Having laid an excellent foundation I was left with the feeling that the book failed to capitalize and deliver the cherry on the top.

The book is highly recommended as one of the new breed of Computer Science books which gives substantial attention to the fundamentals of the subject without becoming bogged down in over-rigorous formality.

Artificial Intelligence and the Design of Expert Systems
by George F Luger & William A Stubblefield, The Benjamin/Cummings Publishing Co., 1989.
Artificial Intelligence: A Knowledge-based Approach
by Morris W Firebaugh, PWS-Kent Publishing Co., 1989.
Reviewer: Prof G D Oosthuizen, University of Pretoria

One of the primary goals of an Honours course is to introduce students to a field in such a way that they arrive at enough insight into relevant issues to enable them to conduct further research on their own. To this end a text book which is used ought to reflect the current view of the field. Because of the rapid expansion of the field of Artificial Intelligence (AI), we have now finally outgrown the era dominated by the books by Winston and Charniak and McDermott. In the past five to ten years much new work has been done, and new insights have been gained. Introducing AI, therefore, requires a marked shift from the previous emphasis on a few historical systems embodying a number of famous methods, to a more generic approach - an approach which highlights those fundamental representation and search models that span all the different application areas and strategies of problem solving. Of course, since AI still does not have a well developed theory, references to seminal systems continues to fulfil an important role.

Both of the above books are good text books, characterised by a balanced coverage of Prolog and Lisp. They also reflect and consolidate much of the work of the past few years done in areas such as knowledge representation, machine learning, the work done under the heading of Expert Systems and even the recent work on neural networks. But the most important feature that they share is the accurate and up to date overall picture of the subject provided; the broad framework for the understanding of AI that is created without neglecting work of historical importance. There are still references to these works, but they are placed in perspective in relation to new developments.

The book of Luger & Stubblefield (L&S) is more language oriented than Firebaugh's book. A characteristic of L&S is that AI approaches to representation are related to the Object Oriented approach. Whereas L&S includes chapters on advanced AI programming techniques in Prolog and Lisp, it does not address pattern recognition, computer vision and robotics. (Firebaugh has chapters on each of these themes.) These omissions are understandable, since AI has diversified so much recently that it is difficult to cover all applications in one book.

If I had to select one of the books, it would be L&S. Although L&S gives poor coverage of Machine Learning, the book's overall presentation is very good. In particular, the chapters are well-organised, and the overall approach to AI - starting with the core aspects of *representation* and *search*, followed by chapters on AI languages - is coherent. The authors also make very good use of graphical representations and illustrations to convey ideas.

Books Received

The following books have been sent to SACJ. Anyone willing to review a book should contact the editor. The book will be sent to him for review, and may be kept provided that a review is received.

- D Bustard, J Elder & J Welsh, [1988], *Concurrent Program Structures*, Prentice-Hall Inc., Englewood Cliffs.
- R Cafolla & A D Kauffman, [1988], *Turbo Prolog Step by Step*, Merrill Publishing Company, Columbus, Ohio.
- S Hekmatpour, [1988], *Introduction to LISP and Symbol Manipulation*, Prentice-Hall Inc., Englewood Cliffs.
- K L Clark & F G McCabe, [1984], *micro-PROLOG: Programming in Logic*, Prentice-Hall Inc., Englewood Cliffs.
- D Crookes, [1988], *Introduction to Programming in Prolog*, Prentice-Hall Inc., Englewood Cliffs.
- M J C Gordon, [1988], *Programming Language Theory and its Implementation*, Prentice-Hall Inc., Englewood Cliffs.
- J G Hughes, [1988], *Database Technology : A software engineering approach*, Prentice-Hall Inc., Englewood Cliffs.
- R Milner, [1989], *Communication and Concurrency*, Prentice-Hall Inc., Englewood Cliffs.
- T J Myers, [1988], *Equations, Models and Programs*, Prentice-Hall, Inc., Englewood Cliffs.
- N C Rowe, [1988], *Artificial Intelligence through Prolog*, Prentice-Hall Inc., Englewood Cliffs.
- D A Protopapas, [1988], *Microcomputer Hardware Design*, Prentice-Hall Inc., Englewood Cliffs.
- H Eisner, [1988], *Computer-aided Systems Engineering*, Prentice-Hall Inc., Englewood Cliffs.
- S H Unger, [1989], *The essence of logic circuits*, Prentice-Hall Inc., Englewood Cliffs.
- R J Young, [1989], *Practical Prolog*, Van Nostrand Reinhold, New York.

NOTES FOR CONTRIBUTORS

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as a Communications or Viewpoints. While English is the preferred language of the journal papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted **in triplicate** to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin. Distinguish clearly between such cases as:
  - upper and lower case letters;
  - the letter O and zero;
  - the letter I and the number one; and
  - the letter K and kappa.
- References should be listed at the end of the text **in alphabetic order** of the (first) author's surname, and should be cited in the text in square brackets. References should thus take the following form:

  [1] E Ashcroft and Z Manna, [1972], The translation of 'GOTO' programs to 'WHILE' programs, *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255.

  [2] C Bohm and G Jacopini, [1966], Flow diagrams, Turing machines and languages with only two formation rules, *Comm. ACM*, **9**, 366-371.

  [3] S Ginsburg, [1966], *Mathematical theory of context free languages,* McGraw Hill, New York.

Manuscripts *accepted* for publication should comply with the above guidelines, and may provided in one of the following three formats:

- in a **typed form** (i.e. suitable for scanning);
- as an **ASCII file** on diskette; or
- in **camera-ready** format.

A page specification is available on request from the editor, for authors wishing to provide camera-ready copies.

Charges

A charge per final page, scaled to reflect scanning, typesetting and reproduction costs, will be levied on papers accepted for publication. The costs per final page are as follows:

Typed format:	R80-00
ASCII format:	R60-00
Camera-ready format :	R20-00

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 1, January 1990
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 1, Januarie 1990
ISSN 1015-7999

Contents