7th SA COMPUTER RESEARCH SYMPOSIUM
PROCEEDINGS

Guest Editor: Judy M Bishop

Organised by the SA Institute of Computer Scientists
in association with the Computer Society of SA

Sponsored by Persetel and the FRD
SPECIAL ISSUE - 7th SA COMPUTER SYMPOSIUM

PREFACE

When the first SA Computer Symposium was held at the CSIR in the early eighties, it was unique. There was no other forum at the time for the presentation of research in computer science. In the intervening decade, conferences, symposia and workshops have sprung up in response to demand, and now there are several successful ventures, some into their third or fourth iteration. Each of these addresses a specific topic - for example, hypermedia, expert systems, parallel processing or formal aspects of computing - and attracts a specialised audience, well versed in the subject and eager to learn more. For the main part, the proceedings are informal, and certainly not archival.

SACRS, though, is still unique, in that it deliberately covers a broad spectrum of research in computing, and in addition, seeks to provide a lasting record of the proceedings. To achieve the second aim, we negotiated with the SA Institute of Computer Scientists for the proceedings to form a special issue of the SA Computer Journal, and the copy you have in front of you is the result. The collaboration between the symposium committee and the journal’s editorial board placed high standards on the refereeing and final presentation of the papers, to the symposium’s benefit, while we were still able to maintain a fresh, audience-oriented approach to the selection of papers.

This is SACJ’s first such special issue, and the largest issue (at 145 pages) to date. We hope that it is only the beginning of future such collaborations.

In all 29 papers were received, all were refereed twice, and 19 were chosen for presentation by the programme committee. All the papers were thoroughly revised by the authors on the basis of the referee’s comments, and the committee’s suggestions aimed at making the material more accessible to a broadly-based audience. Papers had to be new, and not to have been presented elsewhere, a requirement that is still unusual within the SA conference round.

A third goal of SACRS has been to invite keynote speakers, usually from overseas. This year, we are fortunate to present Dr Vinton Cerf, the father of the Internet and a world-renown expert on computer networks. Although his paper is not available for this special issue, it will appear later in SACJ. Through the good offices of Professor Chris Brink of UCT, we also have three other speakers from Germany, Canada and the US adding interest to the event, and two of their papers appear in this issue.

The programme committee originally devised a theme for the symposium - "Computing in the New South Africa". We received several queries as to the meaning of this theme, but unfortunately few papers that addressed it directly. One prospective author went as far as to enquire whether computer research would survive in the new South Africa. Another felt that his work was definitely not in the theme, as it was genuine, old world, basic, theoretical science! Nevertheless, there are two papers that consider one of South Africa’s key issues, that of language. Others look at the success we have achieved in applying technology to mining, and the future of low-cost operating systems. In all, the mix of papers represents a balance between the theoretical and the practical, the past and the future, all firmly based in the computing of the present.

Organising the symposium has involved the hard work of several people, and I would like to thank in particular

• Derrick Kourie, my co-organiser, and the editor of SACJ for his invaluable advice and hard work throughout the planning and implementation stages;
• Rieł Smit, the production editor, for attaining such a high standard in such a short time for so many papers;
• Gerrit Prinsloo and the staff at the CSSA for their efficient and quite delightfully unfussy organisation;
• Persetel for their very generous sponsorship of R25000, and Tim Schumann for taking a genuine interest in our events;
• the Foundation for Research Development for sponsoring Vint Cerf’s visit;
• and finally the Department of Computer Science of the University of Pretoria for providing the ideal working conditions for undertaking ventures of this kind, and especially Roelf van den Heever for his unfailing encouragement and support.

Judy M Bishop
Organising Chairman, SACRS 1992
Guest Editor, SACJ Special Issue
Referees

The journal draws on a wide range of referees. The following were involved in the refereeing of the papers selected for this special issue. Their role in certifying the papers and their contribution to enhancing the quality of papers is sincerely appreciated.

John Barrow  
Ronnie Becker  
Danie Behr  
Sonia Berman  
Liesbeth Botha  
Theo Bothma  
Chris Brink  
Peter Clayton  
Ian Cloete  
Antony Cooper  
Elise Ehlers  
Quintin Gee  
Andy Gravell  
Wendy Hall  
David Harel  
Scott Hazelhurst  
Derrick Kourie  
Willemm Labuschagne  
Doug Laing  
Dave Levy  
Graham McLeod  
Hans Messerschmidt  
Deon Oosthuizen  
Ben Oosthuysen  
Neil Pendock  
D Petkov  
Martin Rennhackkamp  
Cees Roon  
Jan Roos  
John Shochot  
Morris Sloman  
Riel Smit  
Pat Terry  
Walter van den Heever  
Lynn van der Vegte  
Herman Venter  
Herna Viktor

UNISA
University of Cape Town
University of Pretoria
University of Cape Town
University of Pretoria
University of Pretoria
University of Cape Town
Rhodes University
Stellenbosch University
CSIR
Rand Afrikaanse University
DALSIG
University of Southampton
University of Southampton
The Weizmann Institute of Science
University of the Witwatersrand
University of Pretoria
UNISA
ISM
University of Natal
University of Cape Town
University of the Orange Free State
University of Pretoria
University of Pretoria
University of the Witwatersrand
University of Natal
Stellenbosch University
University of Pretoria
University of Pretoria
University of the Witwatersrand
Imperial College, London
University of Cape Town
Rhodes University
University of Pretoria
University of Pretoria
University of Fort Hare
Stellenbosch University
Managing Methods Creatively

Graham Mcleod

Information Systems, Faculty of Commerce, University Of Cape Town, Pvt Bag, Rondebosch, 7700

Abstract

Organizations have used systems planning and development methodologies for some time now. Results are mixed, with some reporting substantial benefits, and others abandoning the process without achieving the desired benefits. The fact that this can occur with the same methodology indicates that there are other factors at work. Research into these factors, and their impact on the success or failure of a particular methodology in a particular situation is now accumulating.

We can learn valuable lessons from these findings to apply to engineering of new methods and management of the methods that we use. This paper explores some of these ideas, and suggests how we can balance many factors which at first appear irreconcilable. The approach includes a unique method model and management approach.

Keywords: Methods, Methodologies, Management, Modelling.

Computing Review Categories: D.2.9; K.6.1

Received February 1992, Accepted April 1992

1. Problems without Methods and Techniques

Some highly skilled practitioners argue that methods are only a crutch for the incompetent and that they should be dispensed with altogether, since they encourage "replacing insight with ritual"

- Downing, 1990

This is true if they are blindly followed without regard for their suitability and without an understanding of the intentions of each step. But there are also problems where there is a lack of methodology:

Lack of Standards leads to a situation where each time a particular task is performed, the person doing it has to invent their own set of techniques and representations. This would be tolerable if every practitioner was an expert in information systems techniques and their applicability to various circumstances. In truth, these skills are extremely rare. Assuming there were sufficient experts available, there would still be problems of lost information and mis-communication where varying techniques and representations were chosen. The organization would be at risk when the experts move on to other roles, leaving an unstructured pot-pouri of great variety, which will require a vast array of skills to interpret.

Lack of standardisation has serious consequences whenever information must change hands. This occurs within a project between persons responsible for different areas (e.g. technical and business aspects), as well as between phases (e.g. between analysis and design). It also impacts communication of the project with user personnel, facilities, I.S. Management and support groups (e.g. Data Base Administration). In organizations where systems are crucial to the business, they will tend over time to become more integrated and will require higher levels of data sharing [51]. Frequently, several development projects will proceed in parallel with the intent that they should integrate smoothly in production. This is impossible unless the respective teams have a common language in which to exchange critical scope, design and interface information.

Poor Quality arises where there are no agreed deliverables from a task (or no agreed tasks!), and thus no performance standards for what must be produced [25, 23]. This allows serious problems to escape detection until very late in the lifecycle, when they are extremely expensive to fix [12]. Good methodologies provide an expected performance standard for every step and product, thus ensuring that quality is built in [22]. They will also normally provide a safety net in the form of regular audit or review points as the product is specified, designed, built and tested [22, 36]. This adds expense to the early phases of the project, but will normally result in savings later in the lifecycle.

Low Productivity. Where there are no standards, or techniques are employed for an insufficient time, there is no chance for staff to move through the requisite learning curves and to reach high levels of output with a known technique or tool. Productivity and quality suffer as a result. Where quality is poor, significant amounts of work will have to be redone, increasing costs and lowering morale [23].

Maintenance Quagmire. A major benefit of using
structured techniques in a formal lifecycle is the production of documentation. This can be in paper or electronic form (e.g. within a repository), but is essential for the successful operation and maintenance of the application [25, 39]. Since some 70% plus of the lifetime costs of the typical application occur in the maintenance area [10, 15, 37], this is of the utmost importance. If this documentation is missing, inaccurate or incomprehensible, maintenance costs can soar. The organization can be left in a position where critical production applications cannot be adapted to meet changing business conditions.

Poor Estimates. Worse than the technical problems caused where no methodology is in place, are the management problems. If there are no prescribed tasks to complete a project, and we do not know what personnel or skills are required when and for how long, it becomes impossible to accurately plan and estimate projects [25, 38]. This makes cost justification farcical, and can lead to the wrong projects being tackled, or expensive project failures [28]. Also, if accurate plans cannot be constructed, then tracking progress is not feasible.

Unprofessional. Where the above situation exists, the Information Systems personnel will give the impression (not incorrectly) of being unprofessional. The rest of the organization, and particularly management, will regard them as a group of technologists without professional discipline or an understanding of the business. The reporting level of I.T. will reflect this, and the function will not be able to fulfill its potential within the organization.

No basis for the use of automated techniques. Recent models by Humphrey and others [39] indicate that there is a maturity progression through which organizations move before they can take advantage of automating the lifecycle (through extensive use of CASE tools). The use of automated tools cannot be successfully undertaken within a culture which has not yet taken standards and methods to heart.

Humphrey’s levels of maturity

<table>
<thead>
<tr>
<th>OPTIMIZED</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Major Quality and quantity improvements</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MANAGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Substantial quality improvements</td>
</tr>
<tr>
<td>• Comprehensive process measurement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REPEATABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Achieved a stable process with repeatable level of statistical control</td>
</tr>
<tr>
<td>Initiate rigorous project management, management oversight and quality assurance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEFINED</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Achieved foundation for major and continuing process</td>
</tr>
<tr>
<td>Establish a process group</td>
</tr>
<tr>
<td>Establish software development process architecture</td>
</tr>
<tr>
<td>Introduce software engineering methods and techniques</td>
</tr>
</tbody>
</table>

Support automatic gathering of process data

Use data to Analyse and modify processes

Establish basic process management to establish quality and cost parameters

Establish a process database

Gather and maintain process data

Assess relative quality of each product and inform management

Figures from Boehm and others confirm that the list of casualties is long.
2. Problems with Methods and Techniques

"Correct" method contingent. The success of a given methodology in a given situation is contingent upon the quality of the methodology (soundness, maturity) as well as upon appropriateness to the application domain, the political climate regarding its use, skills and backgrounds of the I.S. practitioners, effectiveness of training, and attitude of the user community [24, 6].

Technological change is extremely rapid, and accelerating. Methods proven effective in a given technical environment may prove ineffective in the next environment encountered.

Changing Techniques. Techniques specified by the methodologies are evolving, and new ones are constantly emerging. Witness the recent rise of the object oriented paradigm [52].

The learning curve for a new methodology is time-consuming and expensive for an individual. For an organization, it is a major investment and may take several years, since it represents a culture change [14]. Only once significant progress has been made toward mastering the techniques and methodology can benefits be realised.

Tool Immaturity. Powerful CASE tools are available, but the technology is still immature and very few tools provide a totally integrated solution, or support more than one method option or target environment [39, 31].

Methodologies themselves have limited coverage. Some address only planning, others only development or package selection. Some attempt to cover everything, and in the process become unwieldy and impossible to master in totality [31].

3. Conventional Methods

Too Rigorous. Some methods have evolved from rigorous academic or mathematical descriptions of very narrow problems [45]. These are essential to push back the boundaries of what we can tackle with certainty. However, there are significant problems when we take these into practice:

- Few real-world problems are as well defined or "neat" as we would like
- The rigorous, narrow-field methods seldom integrate easily with other necessary techniques within the lifecycle [31]
- Application of the techniques may require skills which the average practitioner does not have

<table>
<thead>
<tr>
<th>PHILOSOPHY</th>
<th>The overall approach, concept and point of departure for all the ensuing activity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD</td>
<td>The &quot;What&quot; - What we must produce, in what sequence, and with what dependencies.</td>
</tr>
<tr>
<td>TECHNIQUES</td>
<td>The &quot;How&quot; - How we achieve the what. Details of techniques to apply, prerequisites, integration.</td>
</tr>
<tr>
<td>TOOLS</td>
<td>Things to help us apply the techniques. Includes CASE, Project Management software, IPSE's, Configuration Management Software, Word Processing etc.</td>
</tr>
</tbody>
</table>

Methods Framework
These techniques are often very domain-specific, and can be easily invalidated with new technologies.

They may require an investment of time and resources which is unacceptable in the commercial sector [45, 14].

**Not rigorous enough.** Conversely, many techniques in common use have no strong theoretical or empirical base, but are merely the collected opinions of their inventors. These are at best successful by force of personality, luck or emotional attachment: at worst, misleading and dangerous. Their attraction is that they offer an apparently simpler, quicker (and hence cheaper/less arduous) way to achieve a desired result. They should not be contemplated for situations where the desired system is critical either to safety or the organization's mission.

**Too Big/Learning Curve.** Very comprehensive methods are large [29]. Some commercial products extend to fifty printed volumes. There is a significant learning curve associated with any methodology. It is unrealistic to expect the developer to learn and have at his disposal several methods from which he can choose one appropriate to the project [6].

**Too trivial/Fragmented.** Simple, easily-comprehended methods are smaller and have a reduced learning curve, but they may have limited applicability. I.e. they may not cover all phases of the lifecycle, or not offer alternatives required, e.g. package selection versus custom development [44, 35].

**Slow evolution.** Most methodologies are either adaptations made by the practicing organization from the work of several authors, or commercial products maintained by their respective vendors [14]. The updating of these products is typically laborious and slow. This leads to delays in the incorporation of important new techniques and concepts. This can lead to frustration amongst the user community who see that they are not keeping up with the state of the art in a particular area [42].

**Supplier Dependence.** Where externally-sourced methodologies are used, the client is dependent upon the supplier for changes and enhancements. Of course, local modifications may be made, but this is problematic, since these may be invalidated by the next "official" release. The organization may find itself in an awkward situation if it has made a significant investment, only to find its own and the supplier's strategies diverging.

**Adaptation Required.** Seldom is an "off the shelf" method entirely suitable for immediate application within the organization [14]. There are always unique management and cultural implications to take into consideration. The extent to which the chosen method permits and facilitates these without a maintenance nightmare is of fundamental importance.

**Integration Difficulties.** Where techniques and tools are chosen from several sources, the integration of these into an in-house total solution is non-trivial. It is also likely to commit the organization to costly ongoing maintenance as the respective products evolve.

Techniques prescribed by methodologies evolve over time. New techniques offer attractive benefits, but significant effort is required to integrate a technique successfully with the rest of the method. For example, object oriented analysis is attractive, but how does one integrate this with entity modeling, and JAD type sessions with users? Achieving a smooth migration is very difficult, and discontinuities can severely impact quality and productivity.

**Tools** are currently maturing to support technical tasks performed [39]. E.g. Function modeling, data modeling, process design etc. Some tools support a single method, which is implicit in the tool. Others support multiple approaches, but rely upon the practitioner to perform the correct steps in the correct sequence [13, 34]. None, to the authors knowledge, contain the method definition explicitly - with the exception of the Virtual Software Factory (VSF), a tool for the generation of CASE tools [47].

VSF allows the definition of method rules to the tool. While this is at a high level, and therefore concise and productive, it still requires unique and scarce skills. There is a need to begin introducing other tools such as group decision support software, to address the less technical aspects of the interactions during the lifecycle. A further development could be the integration of electronic mail into the process, as the literacy of the community increases and access to these facilities becomes commonplace.

**Technical Focus.** Traditional methods concentrate upon the technical aspects of the lifecycle. This author believes that there are now many competent methods which address these issues (e.g. Information Engineering [39], Method/I [1], SSADM [43], Tetrarch [49], Infomet [33] etc). The tougher problems to crack are the management and people aspects. This has been recognised by the development of management umbrellas to the technical methods e.g. GOLD for Tetrarch [22] and Prince for SSADM [43]. This author believes that these "add-on's" should now be fully integrated with the technical aspects into a unified approach. Joint Application Development (JAD) techniques are being widely adopted as a structured way of managing the interaction of various specialists, technical people and line staff within the lifecycle. These also need to be integrated into the process followed, as
4. A Proposed Solution

People Focus. People are ultimately the central factor for consideration. It is people who require, specify, analyse, design, build, test and use applications. We can have the most fantastic technical tools, but we ignore human issues at our peril. If the manager cannot understand the deliverable produced, or the analyst will not use the CASE tool provided, we lose. As systems become more pervasive and critical in organizations (and outside organizations), increasing attention must be paid to social and political implications. "Culture capable" techniques must be used to create "culture compatible" applications which fit the environment and user community. It is already evident that a much greater proportion of development effort will in future be expended on engineering the user interface to products. An example of this is the current re-engineering of products for the Windows/Presentation Manager environment. Fortunately, powerful tools and standards will alleviate this problem fairly quickly.

An interesting phenomenon is the emergence of methodologies which have a very strong people focus, almost to the exclusion of the technical. Multiview, in the U.K., is a good example [5].

Management Focus. The technical problems are largely solved - this is typically not where projects fail today. Most serious problems are encountered in management aspects - Planning, estimating, tracking, control, quality assurance [28]. Additionally, management is called upon to manage an increasing variety of project types: strategic planning, application development, reverse engineering, package implementation, and others [35].

There is a need for a common management approach to these, so that skills can be transferred across project types, and so that reporting and judgement can be consistent. Valuable lessons can be learned here from the Software Engineering and Configuration Management disciplines [32]. These offer a consistent set of phases, quality control points and management process for a variety of project types. Unfortunately, they incur significant overhead unacceptable in the commercial sector. This author believes that we need to make these techniques less resource intensive and educate management to accept the overhead for mission critical projects.

Layered Architecture. Current methods are often poorly structured internally [44]. Like bad systems, they have confused levels of abstraction, high coupling between apparently unrelated areas of the method, and are very difficult to maintain without disturbing things unintentionally. It is a gargantuan and specialised task to integrate components from competing methods. The
situation is analogous to the early days of data communications. As in this field, the solution lies in standards and layered architectures. We need to distinguish between, Philosophies, Methodologies, Techniques and Tools [31]. This allows us to keep the foundations of philosophy and management process stable while the techniques and tools evolve.

Method Model. The methodology needs to be represented and stored in a structured, explicit form. The author has developed a method model for this purpose. This has four inter-related components:

- **Product Model** which specifies structure, relationship and detail for everything we will produce
- **Work Model** which specifies structure, responsibilities and dependencies for everything we will do
- **Organization Model** which specifies the roles, relationships and complements of human resources
- **Infrastructure Model** which specifies environment, facilities, tools, techniques, standards etc available to accomplish the tasks.

The model is explicitly described and includes management as well as technical tasks. It is capable of representing a variety of methods in a consistent way for easy comparison, integration, management and maintenance.

Quality Integration. Quality is an underlying philosophy and cannot be grafted on afterwards. It is implicit in every task performed, and every product produced [23]. Consequently, within the description of the method, every task and deliverable must have associated quality standards. In addition, formal quality review points and an error detection/correction scheme are built into the management process. These apply to the method itself as well as the product being produced by the project.

Dynamic Lifecycle Generation. The full method is large and would be unwieldy if applied on an individual project. Consequently, techniques are provided to allow only tasks and deliverables relevant to the current project to be extracted from the model and specialised. This is achieved without loss of integrity and respecting dependencies. The specialised version then serves as the basis for project planning and management.

Creative Schizophrenia. Organizations need to leverage expertise by using techniques and technologies for
extended periods of time. They also need to come to grips with new technologies, select those that have significant long term benefits and begin learning these early to build expertise and define usage parameters appropriate to their circumstances. This can best be achieved by creative schizophrenia, where the mainstream production teams concentrate on using the current technology effectively and efficiently, while a separate group scans the marketplace, evaluates promising technologies, and advises on adoption and implications for technologies selected.

This group has a dual role - Planning the technological evolution of the organization's infrastructure, and transferring technology and expertise to the production teams. Two local organizations practicing this approach are Old Mutual and Foschini Data.

Self Enhancing. The methodology cannot remain static over time, waiting for the next release. It must evolve and become tuned to the organization, its culture, infrastructure and technology. This can be achieved by having the method explicitly defined in the model, and incorporating method review points into the lifecycle. In this way, projects can take advantage of lessons which were only learnt in the previous phase of a concurrent project.

Consistent Management View. This can be achieved by adopting a Configuration Management framework for the project phases at a high level. This ensures that, at the management level, all projects have a similar look and feel, allowing meaningful comparisons, and accumulation of project management expertise. Detail tasks within the phases will vary dependent upon the type of project, but this is a matter for the project manager and technical staff and will not affect the senior management view.

**Tool Supported.** Ideally, the method definition should reside in a tool. This will allow rapid updating of the definition based upon real-world feedback as well as
Formally can be controlled by the level at which tasks are chosen for estimating, monitoring and quality assurance. The repository is thus a necessity for integration and management of the methods in use. It is also a mechanism to insulate the organization against the turbulence caused by changing techniques and tools.

If the underlying information is represented in the repository then:

- It must be explicit
- It is not lost when practitioners move on
- It is available to a variety of tools
- It can be represented in the format of preference (some people prefer a diagram, others a list)
- It is sharable by multiple users in a controlled fashion
- It is amenable to automated quality assurance checking

Repository technology is still immature, but is definitely central to achieving higher levels of productivity and quality.

5. Conclusion

Key strategies for managing methods creatively which we can summarise from the foregoing are as follows:

- Create a stable method environment, preferably incorporating a method model
- Concentrate effort and attention on the people and management aspects not the technical details
- For production teams, use tried and proven techniques within a framework that allows rapid evolution of the techniques
- Have a separate group scan new technologies and plan the evolution of the technical environment
- Adopt a single comprehensive method definition which is fully integrated and specialise this per project
- If necessary, acquire methods from various sources, but integrate them under the single framework
- Don’t allow deviations from the method - but adapt the method if necessary. Do this in a controlled way
- Look for a repository and tools which will assist the above
- Insist that the method you employ includes quality assurance for every task and deliverable
- Apply quality assurance principles to the projects and to the use of methods
- Adopt a consistent management view of projects based upon the Configuration Management philosophy

Ongoing training is essential in a turbulent environment. We should seek to train to higher levels of skill within the evolutionary framework suggested, rather than periodically re-training the basic skills.

Automation of the technical aspects of the lifecycle will allow more time to be focussed on the people and management aspects. Use of a repository should also ease the integration of tools, facilitate sharing and improve quality assurance.

Quality & Productivity. It is an established fact that higher skills and better techniques lead to higher quality, and that higher quality leads to increased productivity and reduced risk [41]. These are benefits which we can achieve for our organizations by adopting the approaches suggested here.
Bibliography

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as 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 wide margins and 1 ½ or double spacing.
- 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 numbered and titled. Figures should be submitted as original line drawings/printouts, and not photocopies.
- 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 [1, 2, 3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) \LaTeX\ file(s), either on a diskette, or via e-mail/ftp – a \LaTeX\ style file is available from the production editor;
2. In camera-ready format – a detailed page specification is available from the production editor;
3. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

Further instructions on how to reduce page charges can be obtained from the production editor.

4. In a typed form, suitable for scanning.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for \LaTeX\ or camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

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

Proofs

Proofs of accepted papers in categories 3 and 4 above 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 less than 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.

References

## Contents

Forward ................................................................. i  
Referees ........................................................................ ii  

<table>
<thead>
<tr>
<th>Article</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Translation from African Languages to English</td>
<td></td>
</tr>
<tr>
<td>DG Kourie, WJ vd Heever and GD Oosthuizen</td>
<td>1</td>
</tr>
<tr>
<td>Automatically Linking Words and Concepts in an Afrikaans Dictionary</td>
<td></td>
</tr>
<tr>
<td>PZ Theron and I Cloete</td>
<td>9</td>
</tr>
<tr>
<td>A Lattice-Theoretic Model for Relational Database Security</td>
<td>15</td>
</tr>
<tr>
<td>A Melton and S Shenoi</td>
<td></td>
</tr>
<tr>
<td>Network Partitions in Distributed Databases</td>
<td>22</td>
</tr>
<tr>
<td>HL Viktor and MH Rennhackkamp</td>
<td></td>
</tr>
<tr>
<td>A Model for Object-Oriented Databases</td>
<td>27</td>
</tr>
<tr>
<td>MM Brand and PT Wood</td>
<td></td>
</tr>
<tr>
<td>Quantifier Elimination in Second Order Predicate Logic</td>
<td>35</td>
</tr>
<tr>
<td>D Gabbay and HJ Ohlbach</td>
<td></td>
</tr>
<tr>
<td>Animating Neural Network Training</td>
<td>44</td>
</tr>
<tr>
<td>E van der Poel and I Cloete</td>
<td></td>
</tr>
<tr>
<td>HiLOG - a Higher Order Logic Programming Language</td>
<td>53</td>
</tr>
<tr>
<td>RA Paterson-Jones and PT Wood</td>
<td></td>
</tr>
<tr>
<td>ESML - A Validation Language for Concurrent Systems</td>
<td>59</td>
</tr>
<tr>
<td>PJA de Villiers and WC Visser</td>
<td></td>
</tr>
<tr>
<td>Semantic Constructs for a Persistent Programming Language</td>
<td>65</td>
</tr>
<tr>
<td>SB Sparg and S Berman</td>
<td></td>
</tr>
<tr>
<td>The Multiserver Station with Dynamic Concurrency Constraints</td>
<td>75</td>
</tr>
<tr>
<td>CF Kriel and AE Krzesinski</td>
<td></td>
</tr>
<tr>
<td>Mechanizing Execution Sequence Semantics in HOL</td>
<td>81</td>
</tr>
<tr>
<td>G Tredoux</td>
<td></td>
</tr>
<tr>
<td>Statenets - an Alternative Modelling Mechanism for Performance Analysis</td>
<td>87</td>
</tr>
<tr>
<td>L Lewis</td>
<td></td>
</tr>
<tr>
<td>From Batch to Distributed Image Processing: Remote Sensing for Mineral Exploration 1972-1992</td>
<td>95</td>
</tr>
<tr>
<td>N Pendock</td>
<td></td>
</tr>
<tr>
<td>Galileo: Experimenting with Graphical User Interfaces</td>
<td>99</td>
</tr>
<tr>
<td>R Apteker and JM Bishop</td>
<td></td>
</tr>
<tr>
<td>Placing Processes in a Transputer-based Linda Programming Environment</td>
<td>109</td>
</tr>
<tr>
<td>PG Clayton, FK de-Heer-Menlah, EP Wentworth</td>
<td></td>
</tr>
<tr>
<td>Accessing Subroutine Libraries on a Network</td>
<td>117</td>
</tr>
<tr>
<td>PH Greenwood and PH Nash</td>
<td></td>
</tr>
<tr>
<td>A Multi-Tasking Operating System Above MS-DOS</td>
<td>122</td>
</tr>
<tr>
<td>R Foss, GM Rehmet and RC Watkins</td>
<td></td>
</tr>
<tr>
<td>Using Information Systems Methodology to Design an Instructional System</td>
<td>126</td>
</tr>
<tr>
<td>BC O'Donovan</td>
<td></td>
</tr>
<tr>
<td>Managing Methods Creatively</td>
<td>131</td>
</tr>
<tr>
<td>G McLeod</td>
<td></td>
</tr>
<tr>
<td>A General Building Block for Distributed System Management</td>
<td>141</td>
</tr>
<tr>
<td>P Putter and JD Roos</td>
<td></td>
</tr>
</tbody>
</table>