Special Issue: SAICSIT '99
The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

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

World-Wide Web: http://www.cs.up.ac.za/sacj/

Editor
Prof. Derrick G. Kourie
Department of Computer Science
University of Pretoria, Hatfield 0083
dkourie@cs.up.ac.za

Sub-editor: Information Systems
Prof. Niek du Plooy
Department of Informatics
University of Pretoria, Hatfield 0083
nduplooy@econ.up.ac.za

Production Editors
Andries Engelbrecht
Department of Computer Science
University of Pretoria, Hatfield 0083

Herna Viktor
Department of Informatics
University of Pretoria, Hatfield 0083
sacj_production@cs.up.ac.za

Editorial Board
Prof. Judith M. Bishop
University of Pretoria, South Africa
jbishop@cs.up.ac.za

Prof. Richard J. Boland
Case Western University, U.S.A.
boland@spider.cwru.edu

Prof. Trevor D. Crossman
University of Natal, South Africa
crossman@bis.und.ac.za

Prof. Donald. D. Cowan
University of Waterloo, Canada
dcowan@csg.uwaterloo.ca

Prof. Jürg Gutknecht
ETH, Zürich, Switzerland
gutknecht@inf.eth.ch

Prof. R. Nigel Horspool
University of Victoria, Canada
nigelh@csr.csc.uvic.ca

Prof. Fred H. Lochovsky
University of Science and Technology, Hong Kong
fred@cs.ust.hk

Prof. Kalle Lyytinen
University of Jyvaskyla, Finland
kalle@cs.jyu.fi

Dr. Jonathan Miller
University of Cape Town, South Africa
jmiller@geb2.uct.ac.za

Prof. Mary L. Soffa
University of Pittsburgh, U.S.A.
soffa@cs.pitt.edu

Prof. Basie H. von Solms
Rand Afrikaanse Universiteit, South Africa
basie@rkw.rau.ac.za

Subscriptions

<table>
<thead>
<tr>
<th></th>
<th>Annual</th>
<th>Single copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Southern Africa</td>
<td>R80.00</td>
<td>R40.00</td>
</tr>
<tr>
<td>Elsewhere</td>
<td>US$40.00</td>
<td>US$20.00</td>
</tr>
</tbody>
</table>

An additional US$15 per year is charged for airmail outside Southern Africa.

+27 (11) 315-1319 Fax: +27 (11) 315-2276
Contents

Preface
P. Machanick ............................................................... 1

Research Articles
Active Learning: Issues and Challenges for Information Systems and Technology
RD Quilling, GJ Erwin and O Petkova ............................................................... 5

A Generic Modelling Framework for Interactive Authoring Support Environments
Paula Kotze ....................................................................................... 15

O Petkova and JD Roode ....................................................................................... 26

An Information-Theoretic Semantics for Belief Change
T Meyer ....................................................................................... 33

A Complexity Metrics Model for Software Correction
A Törn, T Andersson and K Enholm ....................................................................................... 40

A Conceptual Design for High-Volume Data Processing of Warehouse Database into Multidimensional Database
Paisarn Trakulsuk and Vichit Avatchanakorn ....................................................................................... 49

A Pragmatic Approach to Bitemporal Databases: Conceptualization, Representation and Visualisation
Chiyaba Njovu and WA Gray ....................................................................................... 58

A Building Recognition System
SP Levitt and B Dwolatzky ....................................................................................... 68

Computer Programming and Learning to Write
John Barrow ....................................................................................... 77

Co-operating to Learn using JAD Technologies
TA Thomas ....................................................................................... 87

Critical Success Factors for the Implementation of DSS at a Selection of Organisations in Kwazulu/Natal
URF Averweg and GJ Erwin ....................................................................................... 95

Enhancing the Predictability of Two Popular Software Reliability Growth Models
Peter A Keiller and Thomas A Muzzuchi ....................................................................................... 105
Generalised Unification of Finite Temporal Logic Formulas
Scott Hazelhurst ................................................................. 110

Harmonizing Global Internet Tax: A Collaborative Extranet Model
E Lawrence and B Garner ..................................................... 119

Improving Object Oriented Analysis by Explicit Change Analysis
Lui Yu, Siew Chee Kong, Yi Xun and Miao Yuan ..................... 128

Reconciling the Needs of New Information Systems Graduates and Their Employers in Small, Developed Countries
Rodney Turner and Glenn Lowry ........................................ 136

Shortest Delay Scheduling Algorithm for Lossless Quality Transmission of Stored VBR Video under Limited Bandwidth
Fei Li, Yan Liu, Jack Yiu-Bun Lee and Ishfaq Ahmad .................. 146

Software Croma Keying in an Immersive Virtual Environment
Frans van der Berg and Vahini Lalili ..................................... 155

Some Automata-Theoretic Properties of \( \cap \)-NFA
Lynette van Zijl and Andries PJ van der Walt ......................... 163

Statistical Analysis of an Automated Computer Architecture Learning Environment
JT Waldron, J Horgan and G Keogh ........................................ 168

The CILT Multi-Agent Learning System
Herna L Viktor ................................................................. 176

The Development of a Generic Framework for the Implementation of Cheap, Component-Based Virtual Video-Conferencing System
Soteri Panagou and Shaun Bangay ........................................ 185

The Role of Experience in User Perceptions of Information Technology: An Empirical Examination
Meliha Handzic and Graham Low ........................................... 194

What are Web Sites Used for: Cost Savings, Revenue Generating or Value Creating?
Man-Ying Lee ................................................................. 201

New Ideas Papers
Approaches to Video Transmission over GSM Networks
Bing Du and Anthony Maeder .............................................. 210

From Information Security Baselines to Information Security Profiles
Rossouw von Solms and Helen van der Haar ......................... 215

Grounded Theory Methodology in IS Research: Glaser versus Strauss
J Smit ................................................................. 219

Introducing a Continuum of Abstraction-Led Hierarchical Search Techniques
Robert Zimmer and Robert Holte ......................................... 223
Multimedia as a Positive Force to Leverage Web Marketing, with Particular Reference to the Commercial Sector
Stan Shear ................................................................. 229

Understanding HCI Methodologies
Peter Warren ............................................................. 234

Electronically Published Papers

Experience Papers
A Java Client/Server System for Accessing Arbitrary CANopen Fieldbus Devices via the Internet
Dieter Bühler, Gerd Nusser, Gerhard Gruhler and Wolfgang Küchlin ......................................................... 239

An Object-Oriented Framework for Rapid Client-side Integration of Information Management Systems
Ralf-Dieter Schimkat, Wolfgang Küchlin and Rainer Krautter ................................................................. 244

Distributed Operating Systems: A Study in Applicability
Jürgen Prange and Judith Bishop ................................................. 249

Formal Verification with Natural Language Specifications: Guidelines, Experiments and Lessons so far
Alexander Holt ..................................................................... 253

Introducing Research Methods to Computer Science Honours Students
Vashti Galpin, Scott Hazelhurst, Conrad Mueller and Ian Sanders ......................................................... 258

Visualising Eventuality Structure
ST Rock ........................................................................... 264

Electronically Published Papers
SAICSIT'99

South African Institute of Computer Scientists and Information Technologists

Annual Research Conference 17-19 November 1999

Prepare for the New Millennium

Is there life after y2k?

Mount Amanzi Lodge, Hartebeespoort, South Africa

Sponsors

http://www.cs.wits.ac.za/

PHDS research/PHDS.html

Think different.
Preface

Philip Machanick, Overall Chair: SAICSIT'99

Running SAICSIT'99, the annual research conference of the South African Institute for Computer Scientists and Information Technologists, has been quite an experience.

SAICSIT represents Computer Science and Information Systems academics and professionals, mainly those with an interest in research. When I took over as SAICSIT president at the end of 1998, the conference had not previously been run as an international event. I decided that South African academics had enough international contacts to put together an international programme committee, and a South African conference would be of interest to the rest of the world.

I felt that we could make this transition at relatively low cost, given that we could advertise via mailing lists, and encourage electronic submission of papers (to reduce costs of redistributing papers for review).

The first prediction turned out to be correct, and we were able to put together a strong programme committee.

As a result, we had an unprecedented flood of papers: 100 submitted from 21 countries. As papers started to come in, it became apparent that we needed more reviewers. It was then that the value of the combination of old-fashioned networking (people who know people) and new-fashioned networking (the Internet) became apparent. While the Internet made it possible to convert SAICSIT into an international event at relatively low cost, the unexpected number of papers made it essential to find many additional reviewers on short notice. Without the speed of email to track people down and to distribute papers for review, the review process would have taken weeks longer, and it would have been much more difficult to track down as many new reviewers in so little time.

Even so, the number of referees who were willing to help on short notice was a pleasant surprise. The accepted papers cover an interesting range of subjects, from management-interest Information Systems, to theoretical Computer Science, with subjects including database, Java, temporal logic and implications of e-commerce for tax.

In addition, we were very fortunate in being able to invite the president of the ACM, Barbara Simons as a keynote speaker. Consequently, the programme for SAICSIT'99 should be very interesting to a wide range of participants.

We were only able to find place in the proceedings for 36 papers out of the 100 submitted, of which only 24 are full research papers. While this number of papers is in line with our expectation of how many papers would be accepted in each category, we did not have a hard cut-off on the number of papers, but accepted all papers which were good enough, based on the reviews. Final selection was made by myself as Programme Chair, and Derrick Kourie, as editor of the South African Computer Journal. Additional papers are published via the conference web site.

We believe that we have put together a quality programme, and hope you will agree.

Acknowledgments

I would like to thank the South African Computer Journal production team, Andries Engelbrecht and Herna Viktor, respectively from the Department of Computer Science and Informatics, University of Pretoria, for their work on producing the proceedings. The reviewers listed overleaf did an excellent job: many wrote very detailed reports, sometimes after being called in on very short notice. Inevitably, there were some glitches resulting from the unexpected workload, but the buck stops with the programme chair: I promise to do better next time.

I would also like to thank my own department for putting up with the extra work and expense that running a conference entails. I tried not to burden them with too much extra work, but our secretaries, Zalm Gowar and Leanne Reddy, inevitably had to take on some extra work. John Ostrowick provided valuable assistance with design of our web pages and call for papers poster. Carol Kernick, who handles our finances and membership records, did a fine job of keeping up with the demands of the conference.

Finally, I would like to thank our sponsors, whose contribution made this conference been possible:

- PricewaterhouseCoopers – sponsored generous prizes and the conference banquet
- National Research Foundation (NRF) – provided financial support
- University of the Witwatersrand – provided financial support
- Programme for Highly Dependable Systems, University of the Witwatersrand – provided financial support
- Standard Bank – provided financial support

SART/SACJ, No 24, 1999
Editorial

- Apple Computer – provided equipment for the conference
- Qualica – provided technical support including helping with the conference web site

Web Site

For more information about SAICSIT, including a pointer to the conference site, see <http://www.saicsit.org.za>.

Referees

- Department of Computer Science, University of Pretoria
  - Derrick Kourie
  - Bruce Watson
  - Vali Lalioti
  - Andries Engelbrecht
  - Ivan Mphahlele
- German National Research Center for Information Technology - GMD
  - Gernot Goebbels
- School of Math Stat and IT, Natal University Pietermaritzburg
  - Peter Warren
- Graduate School of Business, University of Cape Town
  - Kurt April
- CSIR
  - James Jardine
- Department of Computer Science, Rhodes University
  - Shaun Bangay
  - Peter Clayton
  - John Ebden
  - Richard Foss
  - George Wells
  - Peter Wentworth
- Information Technology Division, Rhodes University
  - Caro Watkins
- Department of Information Systems, Rhodes University
  - Brenda Mallinson
  - Dave Sewry
- Department of Informatics, University of Pretoria
  - Carina de Villiers
  - Herna Viktor
  - Niek du Plooy
  - Elsie van Rooyen
  - Machdel Matthee
  - Alan Abrahams
  - Jackie Phahlamohlaka
- School of Information Technology & Engineering, University of Ottawa
  - Stan Szpakowicz
  - Dwight Makaroff
- Department of Information Systems, Victoria University
  - Glenn Lowry
  - Peter Shackleton
  - Tas Adam
  - Alastair Wallace
  - Stephen Burgess
  - Dave Burgess
  - Julie Fisher
  - Jerzy Lepa
  - Geoff Sandy
  - Rod Turner
  - Alastair Wallace
  - Andrew Wenn
- Peninsula School of Computing and Information Technology, Monash University
  - Ainslie Ellis
- School of Information Management and Systems, Monash University
  - Angela Carbone
- School of Computer Science & Software Engineering, Monash University
  - Dianne Hagan
  - Judy Sheard
- School of Management Information Systems, Deakin University
  - John Lamp
  - David Mackay
  - Philip Joyce

SACJ/SART, No 24, 1999
• Computer and Automation Research Institute, Hungarian Academy of Sciences
  – Ferenc Vajda
• Department of Computer Science, University of the Witwatersrand
  – Ian Sanders
  – David Lubinsky
  – Conrad Mueller
  – Yinong Chen
  – Bob Baber
  – Vashti Galpin
  – András Salamon
  – Scott Hazelhurst
  – Philip Machanick
  – Zoltán Fazekas
• Department of Electrical Engineering, University of the Witwatersrand
  – Barry Dwolatzky
  – Farzin Aghdasi
• Department of Mathematics, University of the Witwatersrand
  – Jonathan Burgess
• School of Law, University of the Witwatersrand
  – Victoria Bronstein
• Edward Nathan & Friedland Attorneys, Johannesburg
  – Justine White
• Department of Information Systems & Technology, University of Durban-Westville
  – Geoff Erwin
• Department of Computer Science, Clemson University, Clemson, SC
  – Karen Hay
• Logical SA
  – Philip Green
• Department of Computer Science, University of Bristol
  – Alan Chalmers
• Department of Systems and Computer Science, Howard University, Washington, D.C
  – Todd Shurn
  – John Trimble
• Department of Computer Science, University of Cape Town
  – Sonia Berman
• King’s College, London
  – Peter Wood
• Department of Information Systems, University of Cape Town
  – Paul Licker
  – Steve Erlank
  – Alemayehu Molla
• Department of Department of Mathematics and Applied Mathematics, University of Cape Town
  – Ingrid Rewitzky
  – Peter Jipsen
  – Renkuan Guo
• Department of Information Systems, University of the Western Cape
  – Andy Bytheway
• Department of Electrical Engineering and Computer Science, The George Washington University, Washington, DC
  – Rachelle Heller
• School of Biophysical Sciences and Electrical Engineering, Swinburne University of Technology
  – Ian Macdonald
• Electrical and Electronic Systems Engineering, Queensland University of Technology
  – Anthony Maeder
• Computer Science Department, Åbo Akademi
  – Aimo Törn
  – Kaisa Sere
  – Mats Aspnas
• Division of Informatics, University of Edinburgh
  – Graeme Ritchie
  – Arturo Espinosa-Romero
  – Jon Oberlander
Editorial

- Chris Brew
- Alexander Holt

• Department of Computer Science and Engineering, Helsinki University of Technology
  - Tommi Junttila
  - Nisse Husberg

• Technical University of Budapest, Department of Measurement and Information Systems
  - Andras Pataricza

• NASA-Goddard Space Flight Center
  - Nigel Ziyad

• Department of Computer Science & Information Systems, University of Port Elizabeth
  - Janet Wesson
  - Leon Nicholls

• Computer Technology Department, Indiana University Purdue University Indianapolis
  - Tim Price

• Department of Electrical Engineering and Computer Science, The University of Michigan
  - Trevor Mudge

• Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
  - David Forsyth

• School of Computer Science, University of Birmingham
  - Mark Ryan

• Faculty of Mathematics, Computer Science, Physics & Astronomy, University of Amsterdam
  - Carlos Areces

• Department of Computer Science, Universidade do Vale do Rio dos Sinos - UNISINOS Rio Grande do Sul
  - Marcelo Walter

• School of Information Technology & Mathematical Sciences, University of Ballarat
  - Binh Pham
Abstract

The paper proposes a framework for discussing methodologies for developing interactive computer systems. It uses this framework for a sample of systems and shows how certain insights about missing aspects of the methodologies emerge.

Keywords: Human computer interface Methodologies, Software engineering, interactive computer systems

Computing Review Categories: D.2.1, D.2.2

1 Introduction

Usability problems have been cited as a major contributory cause for the slow productivity gains following six trillion US dollars of investment in information technology [4]. If there ever was a software crisis then this is it. Not that there is a shortage of advice as many books, articles and journals pour out a veritable flood of information on human computer interaction (HCI). Much of it is sound, some of it is contradictory but either way it is hard for even a moderately well read practitioner to make sense of all this material. No wonder that Warren and Viljoen [11] (amongst others) found that students have very little idea about how to design an interface. There is a need, therefore, to be able to understand and compare these solutions, and for a more overarching point of view.

Our approach will be to provide a reference model (RM) which can be used to compare solutions. This will allow the various approaches to be placed in a context. The RM model is analogous to the software development life cycle (SDLC) in that it provides a way to reason about how software is created. But unlike the SDLC the RM is not a set of phases, but a number of “objects” that are constructed and this provides another and complementary perspective. The model itself cannot be validated any more than the SDLC or Norman’s stages of interaction [7] can be validated. Nor will it, per se, provide a method of choosing between various methodologies as the RM is neutral.

However, what it does do, is to provide a framework which will allow various writings on HCI to be discussed.

In the rest of the paper we will firstly describe the model and secondly describe how it was applied to a number of approaches in order to gain insight into the scope and techniques used. No doubt some readers will argue that the RM makes distinctions that are not real and others will argue that important boundaries are missing. This is similar to the way the SDLC has been used where, for instance, the object orientated community has argued that the boundary between design and analysis should be blurred. This is not a weakness, it is part of the reason for having a model so these issues can be discussed. The discussion of some sample “methodologies” is done to indicate how the RM can be used to provide an insight into these techniques. It is important to understand that we are not attempting to create a new methodology but a way of analysing and understanding methodologies.

2 The submodels

It is contended that the software development process consists of creating a “model” of the software and its environment, which in turn is decomposed into a number of submodels. We are not going to be very precise about what each submodel contains, rather we will consider them to be containers which might be seen differently by the various methodologies. It might be argued that some of these submodels require further decomposition and that could well be so, but we will argue that to combine any two of them will be harmful to our understanding of the process. Our purpose is to look at the content of what is to be achieved independently of the software engineering processes that can be used to achieve it. We can now look at the individual submodels in more detail.

The psychological model (Psycho) The psychological model embraces all those issues that affect the actual end user interacting with the system. It contains all the material discussed under the umbrella of cognitive psychology, the user’s mental model of the system and even motivational concerns. Cooper [1] even goes as far as to distinguish between the user’s needs on the one hand and desires on the other.

The sociological model (Socio) The sociological model has to do with the broader community that might be impacted by the running of the system. It contains all the material discussed under the umbrella of cognitive psychology, the user’s mental model of the system and even motivational concerns. Cooper [1] even goes as far as to distinguish between the user’s needs on the one hand and desires on the other.
The task model (Task) The task model is concerned with modelling the user's activities. The task model is considered to be independent of the way in which those tasks are carried out in any particular system. Techniques such as task analysis [6], and use case analysis [3] are often used to document the tasks that have to be carried out. The task model does not include any interaction with a computer.

The conceptual model (Concept) Techniques such as use case analysis contain much temporal information, and there is a need for a more abstract view of the services provided by the system. The conceptual model is a view of the system as seen by the user and is often expressed in an object-oriented way.

The developer's model (Develop) The conceptual model discusses what the system should do from the point of view of the person who interacts directly with the system. The developer's model contains much more than that, since developers are concerned with issues such as cost of producing code, the maintainability of that code and the efficiency with which it runs. In a sense the developer's model is concerned with the implementation of the conceptual and interaction models.

The interaction model (Inter) The interaction model is the other side of the system from the task model. It provides the user with a way of manipulating the objects and services provided for by the conceptual model. It is not to be confused with the task model, the task model describes what has to be done, the interaction model implements the user interface issues associated with tasks. In a good system there will be a close relationship between the two, but to see them as identical is to confuse the problem with its solution. Nevertheless the distinction is a fine one, and one can expect some disagreements on the boundary between the two.

2.1 Context

The submodels operate inside a given context that defines other dimensions for the development. These contexts could be thought of as further submodels but a context seems more appropriate as they impinge on virtually all of the other submodels.

Stakeholders The stakeholders are those who could or should influence the development of the software. It includes the various kinds of developers such as interaction designers, analysts and programmers, but also the users and those who might pay for the project. The methodologies will vary depending on which groups are involved in the various stages and submodels.

Software engineering Software engineering (SE) describes the process by which software is created, and thus the order in which the various submodels are visited, and who is consulted. Variations such as waterfall versus spiral, participatory versus developer based, and techniques to create the code are the kinds of issues discussed.

Technology Just as the wider social setting in which a computer system operates is important, so is the technological environment. For instance a photo realistic virtual reality system is not a reasonable design specification using the current kinds of computers found on desktops.

2.2 Combining the submodels

It would be tedious and not very enlightening to discuss all the ways in which the submodels can be combined. Rather some inter combinations are undesirable and we will discuss these.

The conceptual and developer's models Object oriented (OO) programmers contend that these techniques are natural representations of the world [5] and they may well be. However programmers are not only concerned with modelling power but many other issues as well like code reuse, efficiency, simplicity of the programming and so forth. While OO techniques are powerful and useful, the concerns of programmers are very different from the concerns of end users [2]. If the conceptual and developer's models are combined there is a grave danger that the programmer's view of the world will intrude upon the user interface. This is considered by one and all to be an undesirable outcome [7].

The task, conceptual and interaction model Dayton et al. [9] have argued that the conceptual model (which they call the "Task object Model") is only necessary if one wishes to map the essentially procedural task model onto a non-procedural approach, such as object oriented user interface. If the interface maps very closely onto the task model then (they argue) the conceptual model is unnecessary. Even in this case it would seem a mistake to abandon the conceptual model. The conceptual model is necessary to shield the interaction model from the task and developer's models. However it does not necessarily exist in code, and could simply be part of a design document.

3 Interactions

Figure 1 shows the design influences (represented as arrows) at play during the development of a computer system. It is important to distinguish these from information flows and temporal relationships. For instance, information will flow from the developer's model to the conceptual model and thus to the interface in the running system, but we do not show arrows in that direction. We are saying that the design of the developer's model should not influence the conceptual model because to do that would be tantamount to the solution dictating the problem. We indicate an arrow running in the opposite direction, namely from the conceptual to developer's model, indicating design influences.

The diagram emphasises that the system is embedded in a context, namely the technology that enables and constrains the design, the stakeholders who play a role in shaping its form, and finally the process that is used to construct it. Inside this context, the sociological model influences only the task model. The sociological model does not influence the user directly but rather the sociological model influences the tasks the user carries out. This may be a
point of disagreement. It is also noted that the sociological model has no direct influence on the way the interaction takes place with the computer. Such an influence exists but it is indirect. The psychological model, on the other hand, has a direct influence on both the tasks to be carried out, and the way the interface is constructed to carry out those tasks. The task model describes the tasks that the user performs. The conceptual model is an abstraction of those tasks, representing the data and services that the user interacts with. Some information may be lost when the conceptual model is abstracted from the task model, and the task model thus directly influences the interface. Finally the developer's model is determined by (but does not influence) the interaction and the conceptual models.

The main undesirable influence is the interaction model dictating to the task model what is to be done. This would mean "how" it is done dictates "what" is to be done! The other undesirable influence is from the developer's model to anything else. The problem is the same as the above.

4 Some methodologies

The idea of comparing "methodologies" below is to show how the RM can be used to reason about them and their scope. They are not intended to be a representative sample nor is the discussion intended to rank or select a method. We simply draw attention to interesting aspects of the methodologies.

4.1 The naive developer's approach

The main problem with naive developers is the total lack of any planning or structure to their efforts. This results in any and all of the undesirable interactions taking place between the submodels. In particular the developer's model influences all of the rest of the system, as, what is easy to do, is what gets done. Careless rapid prototyping may also suffer from this malaise as eventually the system becomes the definition of the system.

Warren and Viljoen [11] studied a number of students attempting to design a user interface for a video store. What they found was that these designers created an interface directly in terms of tasks that had to be carried out. This in turn led to the one screen per task syndrome, a malady that Cooper [1] calls dialog box pollution. A large number of dialog boxes is undesirable because at each page switch the user losess context making exploratory learning difficult. In RM terms the intermediate conceptual model is missing.

A similar problem can be expected to occur when the developer's model directly influences the interaction model. This particularly manifests itself in the one screen per relation approach which some database management systems tend to encourage. But a relation does not necessarily map onto a concept in the user domain so this technique may not be appropriate.

4.2 Cooper's analysis

Cooper's [1] [2] highly entertaining accounts of what is wrong with modern software can largely be attributed in RM terms to the developer's model influencing the rest of the system. An example is that programmers tend to see each branch with equal importance because they expend the same effort writing the code. The end user might prefer the most frequent branch to be defaulted to avoid making the same choice each time. For instance, it is irksome to have to choose printer parameters every time when they hardly ever change. Cooper also dwells on the programmer's zero, one, many view of the world which again does not match an end user view where 10 items is a very different phenomenon from 1000 items. Cooper further claims that once code starts to be written this code is cast in stone, resulting in the developer's model harmfully influencing the final product if the code has been designed without reference to interface issues. His prescription is to ensure that the developers are only involved in the developer's model and nothing else, and only after the other submodels have been constructed. Cooper also makes an important contribution (or claim) regarding the development of the psychological model. Cooper is concerned about software that attempts to be all things to all people with the inevitable consequence that it is nothing to anybody. In turning Shneiderman’s [8] stricture to accommodate diversity around, Cooper introduces the concept of a “persona” and the software is designed to cater for at most a few persona. The persona are roughly worst case users of their type. He claims that interface designers can visualise such a persona and draw accurate conclusions about their needs without the requirement of abstract psychological models. While Cooper’s writings are worth reading for their enter-
From the point of view of the RM, however, they could not claim to be a complete description of how to design and implement interactive computer systems.

### 4.3 USDP

The unified modelling language (UML) has rapidly established itself as the standard notation for modelling software. An important adjunct is the Unified Software Development Process (USDP)[3] and thus it is interesting to see the extent to which the USDP is in harmony with the RM. USDP is a traditional software engineering methodology in which virtually the only attention given to the user interface is the identification of a "worker" called a "User Interface Designer". USDP is "use case driven" where a use case is defined as "a piece of functionality in the system that gives a user a result of value". The entity making this interaction is called an "actor" and therefore described by the psychological model. However very little seems to be done to model the actors in the USDP. While the USDP claims that use cases have been adopted "almost universally" there are some important critics [5]. Use cases seem to describe actors interacting with what we have called the interaction model. It is not clear that a separate "solution independent" task model is being constructed which could later be mapped on to one of a number of different interaction models. The RM would regard this apparent failure to distinguish between the tasks and the implementation of the tasks, as a weakness. The fact that more than one person at a time may be interacting with the system does not seem to be modelled, nor does the technological context. The situation is summarised in Table 1.

<table>
<thead>
<tr>
<th>USDP</th>
<th>RM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Psychological model</td>
</tr>
<tr>
<td>Use case</td>
<td>Task model / interaction model</td>
</tr>
<tr>
<td>Control + entity class</td>
<td>Developer's model</td>
</tr>
<tr>
<td>Boundary class</td>
<td>Conceptual and interaction models</td>
</tr>
<tr>
<td>Workers</td>
<td>Developers in the stakeholder model</td>
</tr>
<tr>
<td>USDP process</td>
<td>Software engineering model</td>
</tr>
<tr>
<td>Not modelled</td>
<td>Sociological model</td>
</tr>
<tr>
<td>Not modelled</td>
<td>Technological context</td>
</tr>
</tbody>
</table>

### 4.4 Newman and Lamming

An analysis of Newman and Lamming’s (NL) approach is given in the Table 2.

The approach is compatible with the RM but does not really discuss how to construct the final product, but more the process that needs to be adopted to construct it. The RM would predict that the user of the NL approach will be uncertain on how to actually construct the various parts.

### 4.5 The Bridge

The Bridge is, according to its authors [9] "a comprehensive and integrated methodology for quickly designing object-oriented (OO), multi-platform graphical user interfaces (GUIs) that definitely meet user needs". This is a bold claim. It is a proprietary methodology taught by Belcore requiring very definite procedures to be adopted. As this methodology is part of the inspiration for the RN it is not surprising that it is compatible with it. The Bridge deals largely with the issue of creating the task, conceptual and user interaction models, and the processes that are needed to make this happen. Interestingly the developers are involved in the design of the user interface. This is not entirely consistent with the notion that there should be no influence from the developer’s model to the conceptual model or with Cooper’s views.

### 5 Discussion and conclusion

The RM is a framework for discussing methodologies that create interactive computer systems, and its primary purpose is descriptive rather than prescriptive. For instance it allows us to compare who is involved in the design process, in what order various issues are addressed and how the various issues are tackled. However it is also a model that makes certain predictions. It firstly says that if sub-models are not present or two submodels are combined in a given methodology, then that methodology is incomplete. Application of RM has allowed us to understand the nature of naive developers’ problems. It has allowed us to
place the provocative writings of Cooper [1] [2] in context. It can take a standard textbook such as [6] and discuss in what areas the student will be getting advice and in what areas he will not. In this case many of the RM concerns are addressed. It reproduces from an HCI point of view some of the concerns about use cases that Meyer [5] has articulated from an OO point of view and others have found from experience. We contend, therefore, that RM does allow us to reason about HCI methodologies without claiming that other perspectives will not yield other insights or that the RM framework is the only one.

What credence can be placed on the insights that RM gives? It can be judged on its own internal consistency, the predictions it makes about other techniques and whether others find it a useful approach. RM cannot be validated empirically because it is an interpretative view [10] where the models can never be shown to be right or wrong only more useful or less useful. It can only be left to readers to judge if the results are useful to them.

References

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 notes. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as Communications of 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 1½ or double spacing.
- The first page should include:
  - the title (as brief as possible)
  - the author’s initials and surname
  - the 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.
  - 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 according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page, http://www.cs.up.ac.za/sacj, which gives a number of examples.

SACJ is produced using the \LaTeX{} document preparation system, in particular \LaTeX{} 2e. Previous versions were produced using a style file for a much older version of \LaTeX{}, which is no longer supported.

Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

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

Proofs

Proofs of accepted papers may 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.

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 specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.