CONTENTS
SPECIAL 20TH ANNIVERSARY ISSUE
185-189  The Australian Computer Journal: Twenty Years On
         J. LIONS
190-195  The Australian Computer Journal: A Twentieth Anniversary Editorial
         Twenty Deadly Sins Against Professionalism and Progress in
         Australian Computing
         A.Y. MONTGOMERY
196-202  What Could a Client Reasonably Expect a Consultant to do When
         He is Engaged to Assist in the Purchase of a Manufacturing System
         W.D. FERME
203-205  Guidelines for Privacy and Security in Computer Systems
         W.J. CAELLI
206-214  A Checklist of Factors to be Considered when Negotiating
         Software Agreements
         N.F. REIDL
215-218  Non-professional Practices in Computing: Some Thoughts on the
         Next Decade or So
         R.A. COLDWELL
219-221  A Guide to Software Quality for the Developer and Unwary User
         R. FARRELL
222-223  The Computing Professional's Bookshelf
         R. CLARKE
224-230  The Professional Development and Continuing Education
         of Computing Practitioners
         T. WOODINGS

SPECIAL FEATURES
231-236  Book Reviews

Published for Australian Computer Society Incorporated
Registered by Australia Post, Publication No. NBG 1124
A small low-cost system which allows users of UNIX access to the world-wide Tandem On-line transaction processing network.

GOOD CONNECTIONS
The LXN system integrates easily into the Tandem transaction processing network.

HIGH AVAILABILITY
Tandem leads the field in providing On-line transaction processing (OLTP) to UNIX users. A Tandem system supports two mirrored disks and so in the case of any disk failure, the integrity of the data is always maintained. In addition the built-in battery back-up power supply ensures that all data held in memory is also safeguarded.

APPLICATION POWER AND PORTABILITY
Now from any workstation you can run UNIX applications and still access the Tandem OLTP network. Power comes from a 32-bit microprocessor backed by a 1.6 megabyte floppy disk drive, 80 or 170 megabyte hard disk storage and a 60 megabyte streaming cartridge tape drive.

EASY TO EXPAND
As you add users, add processor and memory boards. In a fully configured system, memory can expand to 16 megabytes, with 510 more megabytes of hard disk storage. LXN can support up to 32 users and take a huge workload off your mini or mainframe.

SERVICE IS SO EASY
A test of the entire system is just so simple. Your office workers will soon source any problem. And all key componentry is readily replaced.

EXTEND YOUR TANDEM NETWORK, OR START YOUR OWN
It's easy and it's inexpensive. If you have the need for continuing and current information combined with expandability and price performance, then Tandem technology will prove consistently superior.

It's no wonder then that companies in every major industry are choosing Tandem.

For further information contact Tandem at any of our offices listed below.

UNIX and UNIX System V are trademarks of Bell Laboratories. Ethernet is a trademark of Xerox Corporation. MS-DOS is a trademark of Microsoft Corporation.
The Australian Computer Journal: Twenty Years On

J. Lions†

The Australian Computer Journal is twenty years old in November, 1987. This article reviews the history and achievements of the Journal during its existence.

Keywords and Phrases: editorial committee, publication, typesetting, articles

CR Categories: A.0, K.2, K.7.m

This issue celebrates twenty years of publication for the Australian Computer Journal. The first issue, Volume One, Number One, was published in November, 1967. That volume consisted of five issues published at six monthly intervals, so that, even though every volume since the first has been published within a single calendar year, this is still Volume 19. Twenty years span roughly half the history of modern computing which can be variously reckoned to have begun with Atanoff and Stibitz in the late 1930's, or Aitken with the Harvard Mark I in 1944, or Eckert and Mauchly with the ENIAC in 1947, or Williams and the first Manchester machine in 1948, or Wilkes and the EDSAC in 1949. Not far behind these indeed were Pearcey and Beard at CSIR Radiophysics in Sydney, whose CSIRAC machine was completed in the period 1949-51.

Both the Australian Computer Society and the Journal have had a proud association with our early computer pioneers. Dr Trevor Pearcey was largely responsible for the launching the Journal which occurred under his editorship during his term as president of the Society. Pearcey was the second president and besides editing the Journal's first two issues during its first year, he served again as editor from November, 1973 to March, 1976. The first president of the Society was Professor John Bennett who shares many distinctions, not least of which have been that he was Maurice Wilkes' first graduate student and present at the birth of the EDSAC, and that he has served continuously as an Associate Editor of the Journal throughout its entire twenty years.

The Journal's eight editors are listed in Table 1, and the fourteen members of the Society who have served at various times as members of the Editorial Committee are listed in Table 2. The role of the Editorial Committee is primarily to advise and assist the editor in his task.

Some details of the publication history of the Journal are shown in Table 3. Apart from Volume One with five issues, and Volumes Five, Six, Seven and Eight with three issues each, the standard pattern has been for four issues per volume per year, published in February, May, August and November. The initial printing runs were for about 4,000 copies of each issue, whereas recently the runs have been for about 14,000 copies. Significant changes in printing

Copyright © 1987, Australian Computer Society Inc.

General permission to republish, but not for profit, all or part of this material is granted, provided that the ACJ's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society Inc.

1Department of Computer Science, University of New South Wales, Kensington 2033, Australia. Manuscript received October, 1987; revised November, 1987.
technology have occurred during the last twenty years. Originally ordinary text was set using a linotype machine while mathematical text was set on a monotype machine. Printing was done by letterpress. The photographs and illustrations in the early issues were zinc photoengravings. Now phototypesetting and offset lithographic printing are used exclusively.

The Journal has had two distinct cover styles: the initial, rather severe style, used throughout the first ten volumes with only one exception (March, 1977), features the Journal title and issue information together with the Society's logo in white on a solid 'Chinese Red' background. The original ACS logo with its initials enclosed in a diamond frame was used until the present logo was adopted at the end of 1973. The Table of Contents for each issue appeared both on the back cover and inside on the first 'non-advertising' page as well. For Volume Eleven, the cover colour was revised to 'Royal Blue' with modified lettering and layout. For Volume Twelve, the cover colour changed again, to yellow, and the design was further modified. The present cover style, introduced by Chung Yuen for Volume Thirteen, has the Table of Contents for each issue appearing once only, on the front of a predominantly white cover.

Publication and production of the Journal was ini-
tially undertaken by Australian Trade Publications, a subsidiary of Publicity Press who had printed the proceedings of the Society's conference in 1966. When Publicity Press decided to cease its publishing activities, Keith Collins, one of its employees who had been involved with the production of the Journal from the first issue, elected to form a new company to continue publishing this journal and others. The company took the name Associated Business Publications and continued to operate from the same offices as before in Chippendale until 1981, when a disastrous fire occurred at Publicity Press. This unfortunately destroyed most of the back copies of the Journal and forced ABP to move to its present office in Smail Street, Ultimo, and Publicity Press to transfer its printing activities to Alexandria. Later, in 1983, printing was contracted to Ambassador Press in Granville, and most recently, to MAPS Litho in French's Forest who provide a combined printing and mailing service.

Mr Collins shares with Professor Bennett the distinction of an unbroken association with the Journal. His company is responsible for all typesetting, layout, production, printing, mailing and advertising. He is assisted in this by his brother, Ian Collins, and Mario Corel. The latter, who now does all the page layouts, has also had a long association with the Journal extending back twenty years to the time when he was a hand and machine compositor working with hot metal at Publicity Press.

The Journal's page size has remained the same throughout (Demy Quarto, 215mm × 275 mm). The initial binding style was 'perfect binding' but by Volume Seven the present practice of saddle stitched binding was established, and has been followed since. The first ten volumes were all set in Times Roman type, but with Volume Eleven, a change was made to the Optima typeface, which is a more modern, clearer type style that allows slightly denser type without loss of legibility. This was typeset using an IBM Composer by Margaret Conway, who except for a period when she was beset by RSI, has done almost all the typesetting for the Journal during the past ten years.

A change towards computer typesetting of certain articles was initiated, starting with the November, 1985 issue, for several reasons: the difficulty during the period of Mrs Conway's incapacity in obtaining satisfactory typesetting for the complicated text that the Journal often publishes; the increasing availability of manuscripts in electronic form; the recently increased availability of convenient laser printers and computer typesetters; and the willingness of the University of Wollongong to provide access to their phototypesetter, and of Ross Nealon in particular to provide his valuable and necessary assistance. However, the Optima font was not readily available in this new form initially and so the Journal reverted to using the Times Roman typeface. One step forwards, one step backwards! Since 1986 computer typesetting has been used to a varying degree, reaching a peak with the whole August, 1986 issue. It is now used less, both because Mrs Conway has now recovered, and because, in spite of what has recently been written in the popular press, computer systems for typesetting are still far from perfect for routine use. One unresolved problem at the moment is whether to, and if so, how to preserve the electronic versions of the papers that are now available in this form.

A volume index has been published with the last issue of each volume, either as a separate insert or as the centre pages from where it can be easily removed preparatory to permanent binding. Computer methods have been used to prepare this since 1982, and since 1985, it has been printed directly from a master copy prepared using an Apple Laserwriter.

The cost of the Journal has often been of concern to members, but it has always been produced for the Society inexpensively, mainly because of the large amount of voluntary, unpaid labour provided by the editor and his associates. The initial issue had a 'sticker price' of $1.00, which remained unchanged until July, 1974, when it was increased to $1.50. This was again raised in July, 1977, to $3.50 — a substantial percentage increase in any terms. The November, 1977 issue quotes the annual subscription rate at $10.00. This was raised to $15.00 in February, 1979; to $18.00 in February, 1981; and to $20.00 in November, 1981. Since production costs were not rising so quickly, and since the production costs per member have been closer to $6-$7 per annum, no further increase has been made until this year, when the subscription price was again raised by the ACS Council, now to $25.00 per annum.

Earlier issues carried substantially more advertising than recent ones, as at first there was no competition from the Society's other publications and little from the trade press. The first volume carried an average of twenty advertising pages per issue. The peak was reached in May, 1972, with the equivalent of 24 pages of advertising related to a computer exhibition at the United States Trade Centre, from such well-known companies as Varian Data Machines, Honeywell, NCR, Percy Boyden, Warburton Franki, Philips, Ascota Business Machines, BASF, CSA, CSP Equipment, DEC, Electronic Associates, STC, Datronics, Wang, Taylor's School of Commerce, Olympia International, Information Electronics, RACAL, Sands & McDougall, Associated Pulp and Paper, Rintoul, Fairchild and Datanamics.

Since then, the volume of advertising has diminished steadily. Volume Five averaged ten pages of advertising per issue; Volume Ten, eight pages; but Volume Fifteen, only three pages plus the inside covers. Part of this decline can be attributed to the introduction of the Australian Computer Bulletin beginning in November, 1977 (this would have been...
The Australian Computer Journal: Twenty Years On

Table 3. Some Per Volume Statistics

<table>
<thead>
<tr>
<th>Vol. No.</th>
<th>First Issue</th>
<th>Total Issues</th>
<th>Total Pages</th>
<th>Total Articles</th>
<th>Pages/Article</th>
<th>Other Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nov '67</td>
<td>5</td>
<td>290</td>
<td>41</td>
<td>6.0</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>Feb '70</td>
<td>4</td>
<td>188</td>
<td>29</td>
<td>5.6</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>Feb '71</td>
<td>4</td>
<td>188</td>
<td>21</td>
<td>7.2</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>Feb '72</td>
<td>4</td>
<td>188</td>
<td>22</td>
<td>6.5</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>Feb '73</td>
<td>3</td>
<td>144</td>
<td>17</td>
<td>6.5</td>
<td>33</td>
</tr>
<tr>
<td>6</td>
<td>Mar '74</td>
<td>3</td>
<td>144</td>
<td>15</td>
<td>8.0</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>Mar '75</td>
<td>3</td>
<td>144</td>
<td>17</td>
<td>7.1</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>Mar '76</td>
<td>3</td>
<td>118</td>
<td>15</td>
<td>6.6</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>Mar '77</td>
<td>4</td>
<td>170</td>
<td>27</td>
<td>5.4</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>Feb '78</td>
<td>4</td>
<td>160</td>
<td>18</td>
<td>7.6</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>Feb '79</td>
<td>4</td>
<td>160</td>
<td>22</td>
<td>6.0</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>Feb '80</td>
<td>4</td>
<td>162</td>
<td>20</td>
<td>6.7</td>
<td>29</td>
</tr>
<tr>
<td>13</td>
<td>Feb '81</td>
<td>4</td>
<td>160</td>
<td>25</td>
<td>5.6</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>Feb '82</td>
<td>4</td>
<td>164</td>
<td>22</td>
<td>6.4</td>
<td>24</td>
</tr>
<tr>
<td>15</td>
<td>Feb '83</td>
<td>4</td>
<td>164</td>
<td>21</td>
<td>6.6</td>
<td>25</td>
</tr>
<tr>
<td>16</td>
<td>Feb '84</td>
<td>4</td>
<td>168</td>
<td>20</td>
<td>7.0</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>Feb '85</td>
<td>4</td>
<td>204</td>
<td>28</td>
<td>7.0</td>
<td>40</td>
</tr>
<tr>
<td>18</td>
<td>Feb '86</td>
<td>4</td>
<td>204</td>
<td>23</td>
<td>6.8</td>
<td>37</td>
</tr>
<tr>
<td>19</td>
<td>Feb '87</td>
<td>4</td>
<td>236</td>
<td>29</td>
<td>6.8</td>
<td>39</td>
</tr>
</tbody>
</table>

its tenth anniversary — R.I.P.), and later, Professional Computing. However the Bulletin's influence was more far reaching than just diverting some advertising — it also 'annexed' the Society news and announcements, the Society's annual report, etc.

In his first editorial on page 6 of the first issue, Dr Pearcey wrote:

It is intended that this Journal will be of professional standard and is being established to provide a vehicle for the interchange of knowledge of advances in the theory and practice of computing, data processing, automatic control, communications and allied matters in the computer field, with particular reference to the increasingly important role that computers are assuming in all fields of activity — commerce, industry and government, as well as in science and research. The Journal also aims to provide an informative and authoritative picture of Australian computing activity and development in those fields to the computer world at large.

The Australian Computer Journal will be published twice yearly with the eventual object of becoming a quarterly publication. The content and publication is under the control of an Editorial Committee directly responsible to the Council of the Australian Computer Society. All contributions submitted will be subject to consideration by this Committee in consultation with referees authoritative in their particular fields. A high standard will be required of selected contributions.

Successive editors have all endeavoured to uphold these high standards even though they have not always been appreciated by some of the Society's members. Editors have battled over the years to ensure a steady flow of articles for the Journal. Table 3 gives some statistics on a 'per volume' basis. (Please note that 'Total Pages' excludes advertising pages, the covers and the annual index; 'Total Articles' includes Short Communications; and 'Other Pages' are those used not for articles but for announcements, reports, Letters to the Editor and Book Reviews.)

The most successful device for ensuring the flow of articles seems to be the 'Special Issue' devoted to a particular theme or topic. The first of the Journal's special issues was prepared by Tony Montgomery for March, 1977 to celebrate the Australian Computer Society's Tenth Anniversary. This issue contains several commissioned articles surveying events during the previous ten years and recording the early history of the Society, including:

A More Effective ACS?, by R.S. Northcote;
The Australian Computer Society, by J.M. Bennett;
The Roll of Honour The Australian Computer Society 1966-1976,
Real-Time Systems at the Victorian TAB — An Historical Review, by G. Karoly;
Computers in Operations Research, by D.M. Deighton;
Computing Research in Australia, by C.S. Wallace;
Computers and their Social Implications in the Australian Scene, by A.W. Goldsworthy;
The Major Tasks of Data Processing, by W.N. Holmes; and

It makes interesting reading today in light of the authors' expressed hopes and predictions. Ten years does seem a long time in the computing world!

The second special issue, also initiated by Tony Montgomery but produced by Chung Yuen, appeared in August, 1980 and, in the words of the latter, contained 'specially invited papers from several Australian organizations actively engaged in computing research and development, and is meant to give readers an overview of the contribution Australia is making to computing science and technology'. About 1600 extra copies of this issue were published for distribution overseas and to attendees at the Eighth World Congress held in Melbourne that year.

The first special issue devoted to a technical theme was initiated by Chung Yuen and prepared under the guest editorship of John Lions. The theme was Recent Developments in Computer Networks and it appeared in May, 1981. The innovation was well received, and a firm policy of presenting such issues at a rate of about three every two years has been followed by later editors (John Lions, and now Rob Cook). The other special issues on technical themes have been as follows:

November, 1981, Data Base Management, (A.Y. Montgomery, guest editor)
May, 1982, Software Engineering, (P.C. Poole, guest editor)
February, 1983, Programming Languages, (J.B. Hext, guest editor)
May, 1984, *Recent Advances in Computer Auditing*, (B.J. Garner, guest editor)

In May, 1987, the Journal published for the first time a special selection of papers from a major Australian computing conference — the Tenth Australian Computer Science Conference, held at Deakin University in February, 1987. The Journal was represented on the conference’s selection committee and had full access to referees’ reports before the selection was made and the conference programme determined. To avoid problems with duplicate publication, the selected papers appear only in abstract form in the formal conference proceedings. (Conference attendees did receive the full papers, bound separately.) It is hoped that this sets a precedent that can be followed for other conferences, where papers are only selected for presentation after formal refereeing and review. This applies not only for further Australian Computer Science Conferences but also for the Society’s own Australian Computer Conference and conferences sponsored by others such as the IREE.

The transfer of Society news and announcements away from the Journal to the Bulletin reduced the amount of material for the ‘Other Pages’ listed in Table 3. However their number has remained high since the number of book reviews published has grown over the years (for example, 21, 6, 23, and 101 for Volumes, 3, 8, 13 and 18 respectively). The rapid increase in books reviews reflects the increasing interest from book publishers in the computing field.

Last, but not least, is a word (not the last word!) about the published editorial correspondence: Letters to the Editor and Replies. It has been interesting to review this as a whole and to attempt to detect themes and waves of interest. Exactly 35 out of the first 70 issues carry Letters which suggests a steady, but not overvigorius, response from readers. The first issue carried no letters, but it certainly provoked some for the second issue. D.L. Overheu’s paper *On Intelligence, Intelligent Automata and Robots* received a degree of attention that was not equalled until B. Dwyer’s paper *Should Cobol Support Coroutines?* was published in May, 1985. The early correspondence was mostly concerned with technical matters, and a high proportion came from overseas.

However from November, 1973 to August, 1978, there was an almost ethereal silence as though the readers were asleep, or else had forgotten how to write. Then the flow started up again as though it had never stopped. A theme that has recurred more than once during the past ten years has been a questioning of the Journal’s relevance for data processing professionals, presumed to constitute the majority of the Society’s members. Each of the three editors of the time have defended their charge vigorously. Mr Montgomery’s advice (August, 1979) to one reader is still relevant:

Mr Lakin, if you look at articles in the ACJ of some 10 years ago — most of which were observed by practitioners at the time to be of academic interest only — you will see that [the contents of] those articles in many cases are directly relevant to the mainstream of commercial data processing as it is practised today.

There is not room here for further analysis of this claim, but one supporting example is the publication in May, 1983, of a trilogy of tutorial papers on Prolog. Whatever their opinions then, few people today would maintain that logic programming and expert systems are no longer relevant for DP professionals.

Throughout its existence, the Journal has depended for its vitality on help from many different people. Several of these have already been mentioned above. To these I would like to add members of the ACS National Office (especially Bob Rutledge, Roger Shelley and Alan Kelly), my valued assistant for two years, Trevor Vickers, and also the many authors, referees and book reviewers. Book reviewers and the authors of articles and letters are acknowledged when their contributions are published. A different situation applies to the many referees, who provide great service, quiet and unsung, but who expect, and are expected to, remain anonymous in each individual case. General lists of referees are now published from time to time (e.g., August, 1984, and May, 1987) and will appear again; even so it is hard to convey the extent of their many individual contributions.

The last twenty years have been exciting ones for the computing industry and profession. The next twenty — even the next ten — promise to be equally as exciting. What, I wonder, will be said about the state of the world and of the Australian computing industry and profession when this Journal celebrates its fortieth anniversary in November, 2007?

**Biographical Note**

John Lions is a Fellow of the Australian Computer Society and an Associate Professor of Computer Science at the University of New South Wales. He recently completed five and a half years as editor of the Australian Computer Journal.
The Australian Computer Journal: A Twentieth Anniversary Editorial

Twenty Deadly Sins Against Professionalism and Progress in Australian Computing

A.Y. Montgomery†, Guest Editor

This editorial raises questions about the professionalism of the Australian Computer industry. But only a small proportion of the Australian Computing brotherhood would bother to read it. It will not be read by most practitioners because only a small fraction of the entire industry have bothered to become and remain members of the Computer Society. Membership of the ACS should be a commitment to professionalism itself. The fact that this article and issue are exposing and indicating ways of overcoming all sorts of unprofessional activities is a sign that, despite its efforts, the ACS is not achieving a goal of widespread professional practice. Is this one of the reasons for poor membership overall?

The computer industry in Australia has been expanding apace now for more than twenty years, and where is it headed? Australians are probably the most enthusiastic users of computer technology in the world but definitely one of the least competent creators of that technology. That’s, at least partly, why we now “enjoy” what must be the biggest ratio of imports to exports of high technology products of any developed nation. In 1987, Australia imported $4 billion in information technology products and exported $120 million.

Australians are more interested in “turning a quick buck” by speculation on land, minerals, Sydney harbour bridges, horses, lotteries — or, more recently — fooling others with high technology, than actually exhibiting their creative intellectual strengths and integrity and developing productive enterprise.

In the following we mention 20 cardinal sins of the computer industry as it is practised today. Not the only such sins, but ones which we have observed in our involvement in the industry, frequently as consultants advising litigants.

Can you, in the confessional, truthfully claim that you are guilty of none of these sins?

Most of these sins have two sides to them. The one facet is that shown by the computer product supplier, who has certain duties as a professional person to advise his customer. The obverse is the customer himself (optionally, an employee of the customer or a consultant to the customer) who has a duty to ask questions and to perform certain tasks so as to ensure that the equipment is fit for the purposes intended.

Well now, what of our 20 deadly sins?

1. To supply equipment, either software or hardware, which is known to be unsuitable to the customer’s purpose.

Of course, we’ve all heard the joke:
Question: “What is the difference between the salesman of used cars and the computer salesman?”
Answer: “The used car salesman knows when he’s lying.”

Many a true word was spoken in jest! Unfortunately, it is frequently true that super enthusiastic computer equipment salespersons are not sufficiently competent to realise that their statements are false! Such salespersons are — of course — non-professional; incompetent if you prefer. And one meets such salespeople every day. Such a person may be technically quite competent, but not know how to assess the customer’s real needs.

On the other hand, a purchaser is really tempting fate if he relies upon advice from sales people only, if that enquirer has no clear conception of the full gamut of his needs, both currently and during the life of the purchase. In brief, the purchaser will usually need to have a consultant who will be assumed to have competence to assist the customer, his client, to identify his needs and to match those needs to the capabilities of the preferred equipment.

But what is the obligation of the salesman if the client doesn’t have a consultant? Should the professional salesman seek a pecuniary gain when he knows the product the customer (sucker) is buying does not meet his needs; in other words the customers’ lack of knowledge?

If you were recommending a production control computer to be located five hours drive from a major city, would you specify a (non-duplicated) one disk-drive configuration and having no magnetic tape facility? Does such a recommendation reflect a reasonable assessment of the client’s needs? And should you, as consultant to the client, recommend purchase of such a configuration, largely because the supplier is a big, well-established company?

In his article on implementation of manufacturing systems, Bill Ferme gives us some guidance on the steps to be taken to ensure a successful installation. Consultants can use his guide as a basis of provision of service to clients. Purchasers can use his article as a checklist for measuring the performance of supplier and consultant.

2. To fail to assess a customer’s need adequately so that equipment is installed which must inevitably cause severe disruption to the operation of the enterprise. To purchase on
behalf of your employer, equipment without seeking specialist advice outside your area of expertise.

Assessment of the customer’s need can be a lengthy and costly process. Where does the responsibility rest that such is carried out adequately? Should the supplier at least visit and inspect the site before offering equipment for use by the customer? Surely the site needs to be visited if it is a factory, to identify possible problems with power supply, air conditioning, electrical interference, signalling method, toxic substances, hot/cold/humid environments along cable runs, and so on? Do you insist upon visiting client sites before responding to a customer’s request for proposal? If not, why not?

As the “professional” advising the customer when seeking to purchase a piece of equipment, do you obtain the advice of a consultant with specialist skills? Or copies of tender documents used by companies in similar enterprises? Or copies of standard tender documents for equipment of that kind published by the Australian Standards Association? If you fail to do these things, are you acting professionally? (See Australian Standards Association, 1983).

3. To provide a customer with equipment which is untested and ill-documented. To purchase, on behalf of your employer, equipment for which stringent and careful checking has not been conducted.

There are well established methods for testing that suites of programs that have been modified, continue to perform correctly all the functions provided before the modifications. The established methods require that a test data set be written by the systems analyst and run successfully by the validation team each time before delivery of the product to the client. (This test data set is, of course, an essential part of the total program suite and its associated documentation.)

We must accept that it is difficult to ensure that no spurious result will be generated for data other than the test data set, either before or after modification. Realising this difficulty leads us to specifying very complete test data.

Standards of documentation for software have improved over the years but do you plan for and produce user/operator, and systems programmer documentation of a standard sufficient to make each module easily usable, comprehensible and modifiable? Increasingly, critical pieces of code in pivotal procedures are being subjected to confirmation by formal proof. Although formal proof techniques are in their infancy, we must expect mounting requirement for such verification techniques in the future. Can you perform a formal proof, or even a simple piece of code?

Similarly, there are well established methods for testing hardware following installation and subsequent to any field modifications. Do you make it easy for your customer to know that his equipment is functioning effectively, by providing user-friendly diagnostic procedures?

4. To fail to incorporate adequate controls into information processing systems so that accidental or intentional actions lead to system malfunction or loss of sensitive data.

Flagrantly to ignore controls on your information processing system because observance is too tiresome?

By scrutinising application programs written for complainants in legal cases, I have formed the opinion that, generally, controls in information processing systems don’t exist at all! Or if they do exist, they are inadequate to prevent even simple malfeasance. A simple example will illustrate the point. Consider a file of policies relating to insured persons held by an insurance company. Imagine that there are no counts of policy records, nor any control totals (say of the total annual premium payable or total insured value) computed each time the file is read and written, and maintained (let’s say manually) in a register external to the computer. Further imagine that there are ineffectual controls over the borrowing of magnetic media from the library.

Question: How easy would it be to add bogus records to the file or to change the premium or amount insured for real insured persons?

Answer: Very easy! In one (now famous) case, tens of millions of dollars were paid to holders of bogus policies following the “death” of persons.

Control measures can be implemented in information processing systems which make computer fraud extremely difficult, or impossible. Failing to adhere to control practices and procedures leads to directly facilitating criminal activities.

Unprofessional behaviour such as leaving access codes or access procedures lying around or in easily accessible places, increases the temptation and the capability to commit malfeasance. And would you, as a professional, sell or recommend for purchase an operating system intended for a multi-user, multi-application environment which “echoes back” on the terminal the user password when typed on a keyboard?

Technical Bulletins F2 and F3 of the Institute of Chartered Accountants in Australia (ICA, 1976; ICA, 1971) give some guidance on appropriate levels of precaution to adopt for a range of frequently occurring situations. Failure to incorporate adequate system controls is one hallmark of a non-professional practitioner.

5. To use another person’s software or hardware design and to claim it as one’s own for personal gain. To steal data or other intellectual property which is not properly and legally one’s own, but rather the property of one’s current or former employer or a competitor.

To purchase or use intellectual property without checking ownership thereof.

The issue of intellectual property rights is really only warming up, in our opinion. Increasingly, we will find that product creators have a responsibility to keep secret, copyright or patent valuable intellectual property. Increasingly we will find “information poor” individuals striving to steal such property and to use it for personal gain. The effort (investment) in developing complex information technology products is very high; so long as the effort in stealing the product remains relatively low, individuals
and groups will strive to obtain the product details without paying a commercial price for them. The supplier is obliged to incorporate techniques such as self-destruct "bombs", computer system identification checks, and code encryption to aid safe-keeping of the product; to not do so would be unprofessional. Similarly, transgression of the copyright, patent or trade secret rights of an inventor should not be indulged in or tolerated by a professional. In his article on guidelines for Privacy and Security in Computer Systems, Dr Bill Caelli gives some guidance on the steps professionals will need to take to ensure valuable data and programs are not stolen, breached or corrupted.

The purchaser, on the other hand; if he has the slightest doubt, should he not seek to verify ownership of the property?

6. To make undertakings to provide services which are known to be unachievable. For example, to advise a customer that maintenance facilities with adequate response time to the needs of the enterprise are available when such is known to be not the case.

To fail to make reasonable checks that the services can be performed.

Making such misrepresentations we believe are contrary to the trade practices acts and are therefore likely to lead to litigation. But, when the price ticket is $200-$10,000, is it likely that the customer will litigate? Salespersons of microcomputer systems are frequently guilty of making unachievable undertakings which must be viewed as unprofessional. How many cases of misrepresentation has the Small Claims Tribunal dealt with, we wonder?

But surely the professional purchaser thoroughly checks the veracity of claims, particularly if they are likely to effect crucial aspects of the conduct of the enterprise? Norbert Riedl in his article in this issue gives us some guidance on checks to be made when purchasing software products.

7. To issue to the public, shares in companies for products which are not adequately assessed for their technical feasibility and viability. To recommend the purchase of equity or provision of a loan without thorough technological audit.

We believe that issuing of false prospecti, knowing the details to be false, is punishable by law. But, as many of our entrepreneurs are aware, if you don’t know the prospectus is false you’re probably OK. So, if you have a “good idea” for an information technology product, based upon a prototype which is inadequately tested, you can seek equity subscription from the public. Your idea is plausible and in step with its time. When the company floats off, you see that you are allocated plenty of shares, sell some or most of your shares in the high-priced blush of enthusiasm following the float, make a good killing leaving your investing public with subscription in “vapour ware” — neither software, hardware nor firmware — a product consisting mostly of hot air.

If the product does ever earn income, it’s probably after further injections of capital and very long delays. As we said at the outset, in the old days punters and mugs were sold shares in the Sydney Harbour Bridge: those shares were also highly speculative investments!

The issuer of such prospecti is culpable in our view. But so too is the “expert” who claims to have conducted a thorough audit of the technical aspects of the product, whilst having performed only a cursory investigation of the glossy (sales) documentation!

8. To read, insert, amend or delete records on a data base or software for improper purposes.

To submit false transactions to a system for personal gain.

To use data for purposes other than for its designated statutory intention.

Now we come to the issues about the safekeeping of sensitive programs and data.

Again, there are two major facets to this issue. Those relating to the system designer/implementer, and those relating to the system user.

It is the responsibility of system designers to ensure that necessary and sufficient safeguards are incorporated in the system to prevent unauthorised transactions on the data. It is the responsibility of knowledgeable users of information storage systems to carry out only those transactions for which the user is properly entitled.

Examples of this type of misuse are very prevalent. The most recent was of a statutory betting agency’s employee modifying payout programs on horse racing quinellas. Whenever our miscreant wagered on a quinella he received a payout! No matter which horses he selected!

In his article on Non-Professional Practices, Dr Roger Coldwell provides us with insights and statistics for this and other kinds of malpractice.

9. To provide to a second customer a product created for another and to seek payment as though it were original to the second customer.

To employ contractors and not to have them sign non-disclosure agreements and not to keep secret commercially-valuable procedures and processes from them.

Such an action, in our opinion, need not constitute a transgression if the product prepared for the first client was, by agreement with that client, the property of the consultant.

But consider the consultant who writes programs for a client in industry X, drawing upon the special expertise of a knowledgeable and competent manager in the client’s firm, to create a useful suite of application programs. The consultant, realising he has developed a generally applicable package goes to a competitor in the industry and offers a product to that competitor, largely based upon the exact code written for the first client. Furthermore, the consultant seeks for and receives payment, as though it were his property.

Of course, the first client company who employed the consultant acted unprofessionally if it failed to ensure that a non-disclosure agreement be entered into by the consul-
t. Such an agreement might act as some brake on unprofessional action by the consultant, but not necessarily so.

Probably, also, the second client did not act professionally if no questions were asked about the ownership of the product and how it was that the consultant obtained the expert knowledge of the industry embodied in the package?

10. To fail to inform a customer of the need for adequate maintenance of the equipment provided.

Many salesmen and customers enter into the purchase of computer equipment as though it never breaks down and never has any bugs. Such a posture is unrealistic and dangerous. For a salesman not to emphasise the importance of adequate provision for maintenance and repair of the equipment is nearly as reprehensible as the customer who fails to consider the consequences of breakdown upon the enterprise that will use the equipment.

The article by Roger Farrell indicates how to reduce the level of error in software. But it is usually unrealistic to expect zero errors in any software, even that which has been on the market for two or three years.

What action will you take when the equipment breaks down and you have made no arrangements allowing for this possibility? Do you as a salesman, if pressed, admit that there are bugs in the hardware and software and give guidance to your customer on actions to be taken when those bugs become apparent and effect the enterprise’s operations?

11. To fail to inform a customer of the need to make provision for loss of data, the effects of fire, flood and other environmental phenomena upon the ability of the system to perform satisfactorily.

A failing of a similar kind to that preceding. If you are either the supplier or acting as adviser to your employer or a client, you should provide necessary and sufficient data to allow the customer to make proper arrangements to counter these possibilities. Special insurance, water and fireproof safes, regular archiving procedures, off-site storage of duplicates, off-site stand-by facilities? Some or all of these may be essential to the successful operation of an installation and should be the subject of discussion by both professional purchaser and salesperson.

12. To fail to inform the customer fully of the inadequacies of the equipment supplied and the implications for the conduct of the enterprise.

Transgression of this principle cannot be seen as so nefarious as many of the others. Here one issue is how can the salesperson be expected to anticipate non-standard modes of use of the equipment by every customer?

But, consider a salesman who is providing a system intended for multi-user, multi-application transaction processing. Should the salesman indicate that the standard operating system provided with the equipment does not provide security (authorisation) facilities appropriate to the client’s needs? The consultant, if acting responsibly, should be aware of the likely needs of such a client and evaluate the sales proposal accordingly.

Another example. Should the salesperson inform clients that connection of terminals by the RS232C method, will be error prone in electrically noisy environments, particularly for long cable runs? Should the consultant be aware of this and specify a signalling method appropriate to the expected ambient electrical noise?

13. To advise a customer that a complex information processing system can be operated successfully without fully trained staff, when this is known not to be the case.

Consider a country computer facility which intends to run, for the first time, a full computer-based production planning and control system. Is it reasonable for a salesperson to recommend that such an installation could be run without any staff having at least a level of knowledge equivalent to that of a Member of the Australian Computer Society?

As a consultant, would you recommend that such an installation be undertaken and operated without a single professional person?

14. Do you offer your services as an “independent” consultant whilst having a commercial relationship with some suppliers?

If an “independent” consultant acts as an agent for one of the contending suppliers, must there not arise a conflict of interest?

Perhaps clients employing consultants should ask them to formally declare that they have no financial interest in any of the contending suppliers. Such a declaration will not stop a real miscreant, but will cause most people to pause and consider the ethics of their situation.

15. Do you fail to assess accurately the technological risk of a new project; the staff learning time for new methods; the likelihood of problems and pitfalls in new techniques?

If your program production shop is one with hundreds of man-hours of experience in estimating, planning and writing Cobol applications, should you use the Cobol productivity figures (say in successfully tested lines of code per day) in planning for using a 4GL? As an implementor or consultant did you believe the claims made for 4GL’s when they first emerged in about 1982? Are you aware of the problems and pitfalls still evident in many 4GL’s today?

What about today’s expert systems shells? Do they provide all the facilities claimed really effectively? Is it reasonable to expect that they do provide such facilities in clear, unambiguous, user friendly and bug-free ways, immediately after launch on the market place?

In our opinion, providers of products would do well to recognise that newly launched products and methods are likely to be deficient or difficult and to supply early release products only to customers sufficiently experienced to counter problems. Consultants should, for their part, insist upon rigorous benchmark testing, complete documenta-
familiarity with new techniques, without indicating to your client the novelty of the method to you and the risks attached thereto?

Of course, if your client is accustomed to (and even relishes) being used in this way and is made aware of the circumstances, there should be no ethical problem. Indeed, in our opinion, many Australian enterprises are insufficiently willing and prepared to suffer the vicissitudes of a newly launched product, or new methodology. Few large firms and government organisations, which probably have the skilled resources to undertake such a beta-test site role, are prepared to make such a commitment. Again, a sophistication of technical knowledge and accurate risk assessment seems to be lacking here. But without the help of such testing facilities how can Australian products become thoroughly tested and therefore internationally competitive?

Are you the type of adventurous, nationalistic, technically competent Australian who will only employ information technology products which are imported by long-established overseas companies? Or can you accurately assess the developmental stage of a new technique and make appropriate recommendations? Do you do so?

17. When responsible for the implementation of a project, do you wait until the last possible moment to tell your client that the project will be late and over-run budget?

Do you, as a client for an information technology product, not have proper planned milestones for review of progress and insist upon use of proper resource and manpower planning techniques and procedures?

There are well established computer-supported techniques and 'methodologies' for planning and executing projects. Professional implementors and customers use such techniques. Non-professional ones believe the original estimates, don't carefully track progress and frequently are the cause of over-runs.

A related problem is that of obtaining 'incremental commitment' by originally furnishing resource estimates which are known to be low. Once the client is hooked and has expended funds to commence the project, desire to not lose money or "face" leads to further expenditure, and so on.

As a customer in these circumstances, do you check carefully all estimates; query the basis of estimates given you and the control mechanisms to be employed during implementation; do you cut your losses or transfer the project to another implementor if over-runs are encountered?

18. To create software which does not adhere to the relevant standards which define reasonable performance, testing, documentation, ease of modification and maintainability.

To design systems with totally unsatisfactory user interface (e.g. accounts for $0.00, no dates or titles on reports, truncation of surname or address because of inadequate field length specification, inconsistent data names without explanatory glossary, not adhering to programming standards such as structured coding techniques).

There are standards such as ANSI/IEEE Std 829-1983 Software Test Documentation (ANSI, 1983) and ANSI/IEEE Std 830-1984 (ANSI, 1984) Guide to Software Requirements Specifications which professional practitioners should be aware of and use. And again, reference to long-established text-books and to a panel of users should eliminate gross user interface stupidities such as we have thrust upon us every day in the guise of computer supported services. The list of text books published in this issue and developed by Roger Clarke and his colleagues cannot really be claimed as standards. But they give excellent guidance to good practice. How many of these books are on your shelves?

19. Not to seek regularly for and undergo additional refresher and update training in order to retain continuing professional competence.

The ACS is putting the finishing touches onto a series steps that go towards providing ACS Members with mechanisms for ensuring their continuing professional competence. Such competence frequently has two major components: technical and managerial. The ACS PD programme seeks to provide members with improved performance in each of these areas. And in this issue, Terry Woodings provides detailed recommendations for a Professional Development programme in Western Australia.

At this time the overall intention of the ACS is that members be encouraged to remain up-to-date. In the future, the strong possibility exists that Membership of the ACS will require evidence of continued active involvement and training to be furnished, just as is already required with the Australian Society of Accountants.

What steps have you taken to plan for your future continued education? Are you a member of a special interest group, a regular attendee at monthly meetings, a frequent participant at professional development programme activities? As an employer, do you ensure that 10% of your staff time is spent in keeping them technically aware and competent? Confronting a knowledge redoubling period of five years, persons in the information technology discipline are certain to lose the race unless such a level of commitment is embraced.

20. Not to be familiar with the Australian Computer Society Code of Ethics and other Codes of good practice.


The former document spells out, in general terms, the responsibilities of an ACS member to the public, employer and the ACS.
The latter document has sections introducing a code of good practice, indicating the method of work of a data processing professional and suggesting basic steps to be considered in acquiring a computer, and offers a code of good practice.

Some aspects of these codes are now out of date and need modernising. We believe the Australian Computer Society should revamp these codes and generate some new ones.

Nevertheless, many of the recommendations of that 1975 document are still relevant. Are you familiar with them?

In addition, the ACS has published a Code of Good Practice in the Privacy, Security and Integrity of Data (ACS, 1980). Are you familiar with this code, which needs little modernising to be useful?

**SUMMARY**

There are many practices in the design, construction, implementation, testing, sale and use of information technology products which are reflective of non-professional training, education and behaviour. It is sometimes difficult for a practitioner to act ethically, professionally and sensibly. It is time for the ACS to develop and promulgate codes of good practice addressing all of these matters and others you will think of, but we have not. It is also time that the ACS regularly notified members of new Standards as they are endorsed by the Australian Standards Association to take a pro-active role in creating and promulgating Standards of Good Practice.

It is time for you to plan for ensuring your continued professional competence. Through special interest groups, professional development workshops, monthly meetings, state and national conferences, or by contributing to the formulation and application of Codes of Good Practice and Professionalism.

The articles in this issue seek to define proper — professional — practice. The authors may not focus on all the aspects of proper conduct in the situations they address; they were commissioned to write their articles in limited time and they have made a start in the right direction. It may be a pious and naive hope but others of you, in reading these articles, may seek to follow their lead and develop standards of good practice which improve upon their good start. You may care to send your contributions to the ACJ so that others can benefit from your thoughts?

**REFERENCES**


What Could a Client Reasonably Expect a Consultant to do When He is Engaged to Assist in the Purchase of a Manufacturing System

W.D. Fermeng

The purpose of this paper is to give some guidance to managers of manufacturing organisations that are considering the introduction of computer-based information systems. By highlighting the major problem areas and providing a number of checklists of optional solutions, the manager should gain some insight into the processes that should be followed by a consultant giving advice in this area.

Keywords and Phrases: Manufacturing information systems, manufacturing requirements planning, consultant advice, reasonable client expectation.


1. INTRODUCTION

Let us assume that the company is a first time user, and has engaged a consultant to define the company's requirements for an MRP system, identify products which can satisfy the requirements and to recommend a selection from those products of the one most suitable for the company. The paper will follow the consultant through the process and detail reasonable client expectations of his performance.

2. ANALYSIS STAGE

First of all, the consultant must be briefed by the Chief Executive Officer (CEO) concerning the consultant's terms of reference, the CEO's perceptions of the project, the end results expected, possible problems anticipated in both selection and implementation, the company political situation and the future direction of the company and its strategic objectives. The CEO needs to make his expectations quite clear from the outset and to ensure that both parties share the goals of the consultancy work to be carried out.

This briefing would be followed by interviews with all the functional managers. These interviews will either confirm the CEO's viewpoint or contradict it. Either way, the consultant will get a feel for the political climate in the company as well as the potential acceptance of the system across the functions. By this stage, the consultant will have been allocated an office for the duration of the project.

Next the consultant should determine the company's information flow. This is achieved by following the manufacturing cycle commencing at the beginning with the sales order intake in the marketing or sales functions. The consultant interviews the marketing people, at the same time drawing up a system flowchart. He (or she) will gather examples of the documentation and the information volumes and distribution to other functional areas.
Once the consultant has finished, he should retire to his office and construct a diagram of the information flows. This will enable the logic to be examined to check whether any anomalies exist. The reason for this is that most company personnel, including managers, forget to describe some of the systems they use, or are unaware of some practices that have developed. The consultant clarifies his diagram by confirming it with the department concerned, and repeats the procedure moving through the company. For a medium sized company, this can take about a week in total. The product is a flowchart annotated with information volumes as shown in Figure 1. The consultant will also have gathered a working folder containing all the relevant documents used in the flow.

Throughout the process, the consultant will have picked up pieces of information concerning the political climate, attitudes to computerisation, problems endemic to the company, potential implementation problems, established viewpoints on future hardware and software combinations, and a feel for the competence of the management team. Hopefully, by this stage the consultant would have identified a potential company project manager.

The next stage will be the determination of the material flow throughout the company. An example is shown in Figure 2. The consultant follows the flow of materials through the manufacturing cycle starting at the point the material enters the company either as raw material or as a finished part or assembly. By following the material flow,
the consultant gains an impression of the number of computer displays required, as well as indications of the system requirements. For example, if the company is a jobbing shop, there will be a picking or parts marshalling area in between the component manufacturer and the assembly. The consultant will discover whether there is a finished goods warehouse, and if so, how the goods are distributed; over the counter, in the warehouse, or by other means. The material flow also exposes the company's ability to control the flow of material through the plant and whether there is a considerable volume of work in progress throughout the plant. Discussions with the shop floor foreman elicits attitudes to computerisation; will a system be used effectively if it is installed or will it merely be tolerated?

At the end of this stage, the consultant will have identified whether the manufacturing mode of the company is jobbing, batch manufacturing, production flow line (repetitive manufacture), process or a combination of these. The categorisation of the mode of manufacture is important as modern packaged manufacturing systems cater for vertical markets and concentrate on individual manufacturing modes.

A further task at the analysis stage is the company performance measurement. There are many possible measurements, but a few well-known ones are:

**Delivery Performance**
Usually available from the sales or marketing departments. Information about lateness can be well represented by a histogram; see Figure 3.

**Inventory Turnover Rate**
Varies according to the manufacturing mode. Usually available from the financial controller.

**Number of Daily Shortages**
The distribution of shortages should be defined; whether they are vendor shortages, manufactured shortages from the component manufacturing area or raw material stock shortages.

**Scheduling Performance**
A sample of works orders from the shop floor will reveal how many are running late, early or on time. This is a critical measure.

The above measurements will reveal the operating performance of the company and act as a guide to potential usage of a manufacturing system with regard to the solution of problems identified.

### 3. PREPARATION OF SYSTEM AND HARDWARE REQUIREMENTS

By this stage, the consultant should be in a position to prepare a document detailing the system requirements, with a view to issuing a request for proposals from potential suppliers. The specification document should define:
- system overview
- mode of manufacture, to define the type of software required
- modules
- features of each module
- functions of each feature within the modules
- details of screens, reports and on-line jobs
- technical software requirements
- hardware configuration
- record volumes
- support requirements
- implementation problems expected

Manufacturing system modules can be chosen from:
- sales order processing
- bill of materials
- master scheduling
- production control
- financials
- inventory control
- MRP
- costing
- purchasing
- estimation and quoting

Some of the features of an inventory control module might be:
- inventory recording
- ABC analysis
- inventory status
— cyclic inventory
— production control
— production scheduling
— performance reporting
— production expediting
— capacity requirements.

In turn, the *production scheduling* feature might have as its functions:
— scheduled routings
— finite or infinite scheduling
— reporting
— works order creation
— rescheduling
— documentation

The different fields in each screen and report can be defined if necessary. The consultant must use his common-sense as the complexity of these systems is awesome, and the consultant should avoid producing a requirement document as thick as a bible. The goal is to provide the vendor with enough information on the company’s requirements to enable them to respond suitably with system prices and costings for any modifications which might be required. All special requirements like *lot traceability* and *by-products bills of material* should be identified.

The technical software requirements would include the use of a database management system (DBMS), if one is needed, together with any indication of whether a relational, network or hierarchical architecture is preferred. They should also indicate the need for a data dictionary and for productivity tools such as fourth generation languages (4GLs).

Initial estimates of the computer’s configuration should include details of the expected performance, the amount of disk storage predicted, the number of displays required and the number, type and speed of printers. An indication of the back-up requirement should also be included. A list of all the major record volumes (see Figure 4 for an example) will assist the vendor to size the proposed computer according to their software package design.

Other information requested should include:
— Availability of software source code.
— Type of documentation, including user manuals, system manuals, programming manuals, test guides, educational documents and implementation manuals.
— The level of software support and whether it is available in the city of installation.
— Training available in the city of installation.
— Existing software users in the city of installation.

A document prepared containing the above specifications and requests for information would be called a Request for Proposals (RFP).

### 4. AVAILABLE SOFTWARE PACKAGES

A consultant would be expected to keep himself abreast of developments in the field of available software packages and their Australian vendors. To put this into perspective, there are over 70 packages available to an Australian manufacturing company (listed in Figure 5). They have been sold here since the early 1970s, but by 1987 only 700 out of a possible 40,000 manufacturing companies in Australia have acquired systems. How do the vendors make a living? They frequently fail to, and the consultant must be aware of the constant turnover.

Another consideration is that MRP is no longer a stand-alone application; it is now an element of the CIM concept. CIM consists of an array of technologies under four major headings, Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Integrating Technologies and Office Automation (OA). One of the most important recent developments occurred in 1984 with the General Motors initiative to enforce standardisation on computer equipment vendors by setting up a taskforce to develop the Manufacturing Automation Protocol (MAP) for data communications between computers. By December 1986 it was possible to demonstrate communication using a MAP Local Area Network (LAN) between many of the major hardware and software manufacturers’ products connected on a single network. Companies considering the implementation of a manufacturing package cannot ignore the future of the CIM and MAP technologies. All requirements documents should include a requirement for the vendor to commit to support MAP in the future and to provide a route for the system to be acquired to interface via MAP.

To initiate the software selection process, the consultant should recommend sending the RFP to a short-list of selected vendors, perhaps six at most.

### 5. COST/BENEFIT JUSTIFICATION

Once the responses to the RFP have been received, the consultant should prepare a cost/benefit justification. The

---

**Figure 4.**

<table>
<thead>
<tr>
<th>COMPANY'S VOLUMES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Employees</td>
<td>200</td>
</tr>
<tr>
<td>Number of Locations Vic/NSW/Qld</td>
<td>6</td>
</tr>
<tr>
<td>Sales Orders/Consignments Notes/Export Orders per day</td>
<td>250</td>
</tr>
<tr>
<td>Number of Sales Orders etc open</td>
<td>500</td>
</tr>
<tr>
<td>Number of Customers with Accounts</td>
<td>3500</td>
</tr>
<tr>
<td>Number of Finished Goods Items/Parts/Accessories</td>
<td>20,000</td>
</tr>
<tr>
<td>Number of Invoices per month</td>
<td>4,000</td>
</tr>
<tr>
<td>Number of Open Invoices</td>
<td>2,000</td>
</tr>
<tr>
<td>Back Order Items in Existence</td>
<td>100</td>
</tr>
<tr>
<td>Credit Notes per week</td>
<td>30</td>
</tr>
<tr>
<td>Number of Components and Sub-Assemblies</td>
<td>12,000</td>
</tr>
<tr>
<td>Number of Raw Materials/Purchased Parts</td>
<td>16,000</td>
</tr>
<tr>
<td>Number of Open Works Orders: Including Assembly</td>
<td>1,500</td>
</tr>
<tr>
<td>Number of Work Centres (inc. Sub-Contractors)</td>
<td>50</td>
</tr>
<tr>
<td>Number of Operations</td>
<td>20,000</td>
</tr>
<tr>
<td>Number of Tools and Fixtures</td>
<td>10,000</td>
</tr>
<tr>
<td>Number of Shop Floor Labour Bookings per Day</td>
<td>100</td>
</tr>
<tr>
<td>Number of Bill-of-Material Structures</td>
<td>40,000</td>
</tr>
<tr>
<td>Number of Engineering Changes per Week</td>
<td>4</td>
</tr>
<tr>
<td>Maximum Amount of Bill-of-Material Levels</td>
<td>12</td>
</tr>
<tr>
<td>Number of Purchase Orders Open</td>
<td>1,500</td>
</tr>
<tr>
<td>Number of Vendors (Active)</td>
<td>500</td>
</tr>
<tr>
<td>Number of Goods Received Notes per Week</td>
<td>100</td>
</tr>
<tr>
<td>Number of Purchase Orders Created per Week</td>
<td>150</td>
</tr>
<tr>
<td>Number of Operations per Part</td>
<td>7</td>
</tr>
<tr>
<td>Number of Product Families</td>
<td>300</td>
</tr>
</tbody>
</table>
## Consultant Assisted Purchase of System

**Market Position of Available Packages (October 1987)**

<table>
<thead>
<tr>
<th>Company Size</th>
<th>EDP Manufacturing Control</th>
<th>Software Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main-frame</td>
<td>Hoskyns Mas (Facom/IBM)</td>
<td>Cullinet Repetitive MS</td>
</tr>
<tr>
<td>Computer</td>
<td>Cincom Control</td>
<td>SMT OPT</td>
</tr>
<tr>
<td>Market (No. of Companies)</td>
<td>Honeywell HMS</td>
<td>MSA MSAM/COMSERV</td>
</tr>
<tr>
<td>(60) over 1000 employees</td>
<td>Norfolk Business Manager</td>
<td>IBM Copics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Computer Power Mitrol</td>
</tr>
<tr>
<td>Mini-Computer</td>
<td>Hoskyns M (HP/DEC)</td>
<td>Caple Bryce Prism</td>
</tr>
<tr>
<td>Market (No. of Companies)</td>
<td>Data Power DIMAS</td>
<td>ICL OMAC 29</td>
</tr>
<tr>
<td>(1895) Over 500 Employees</td>
<td>Arthur Anderson MAC-PAC/38</td>
<td>Syndicated S/W Appman</td>
</tr>
<tr>
<td></td>
<td>IBM/38 MAPICS II</td>
<td>DATEC WASP-M</td>
</tr>
<tr>
<td></td>
<td>GEC MAXCIM</td>
<td>CBL Command</td>
</tr>
<tr>
<td></td>
<td>Framework MACS</td>
<td>CINCOM MRPS (Prime)</td>
</tr>
<tr>
<td></td>
<td>NCR Command 2000</td>
<td>Praxa Starfire II</td>
</tr>
<tr>
<td>Over 100 Employees</td>
<td>ABS Solutions Apparel BS</td>
<td>Windhover Unibis</td>
</tr>
<tr>
<td></td>
<td>Ultimate MMC/MANFACT II</td>
<td>Hyperscope Pyramid</td>
</tr>
<tr>
<td></td>
<td>HPMM/3000/Jobscope/JIT</td>
<td>SSG Recipe</td>
</tr>
<tr>
<td></td>
<td>IBM S/36 MAPICS II</td>
<td>ICL Safes</td>
</tr>
<tr>
<td></td>
<td>NCR IMCS II</td>
<td>Quantel OMRP</td>
</tr>
<tr>
<td></td>
<td>Computec MAN-FACT II</td>
<td>Wang FACT</td>
</tr>
<tr>
<td></td>
<td>Nixdorf Comet</td>
<td>Horton, Brinkman TMAS/Jobmate</td>
</tr>
<tr>
<td></td>
<td>Taubert MRPS 36/38/P</td>
<td>CBS MITEC</td>
</tr>
<tr>
<td></td>
<td>Momentum Computing PACS</td>
<td>Clegg Driscoll On-line</td>
</tr>
<tr>
<td>Micro-Computer</td>
<td>Mentat MMS</td>
<td>Mayne Nickless IMPCON</td>
</tr>
<tr>
<td>Market (No. of Companies)</td>
<td>Pickpower Manage-2000</td>
<td>Interlink BPCS</td>
</tr>
<tr>
<td>(25513) Over 50 Employees</td>
<td>SCOT (Manf Eng) MICROSS</td>
<td>UNISYS MTMS</td>
</tr>
<tr>
<td>No. of Companies</td>
<td>Scientia MCBA</td>
<td>Albany Jobcode</td>
</tr>
<tr>
<td>1845</td>
<td>Trite K Mate</td>
<td>Honeywell HDMS</td>
</tr>
<tr>
<td>Over 4 up to 50 Employees</td>
<td>CDP Syman</td>
<td>Sea ADAM-PMS</td>
</tr>
<tr>
<td>(No. of Companies)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23668 (86% of market)</td>
<td>Intelec Comics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMS INMASS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interactive Charter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.

Costs are generally relatively easy to determine, but benefits are much more difficult to evaluate. However, these expensive systems must be justified. If the consultant cannot justify it, do not proceed with the project.

A starting-point for estimating the benefits is the definition of a company's current problems, and by estimating the MRP system's impact on the problems. The Australian marketplace is one of the toughest in the world with its own unique problems of relatively small domestic volumes and a limited choice of local vendors. The consultant will have listed the company's major operating problems during the analysis stage, and it is worthwhile to review Figure 6 for a list of typical problems.

In general, the hardware and software costs include:
- hardware
- hardware maintenance contract
- office alteration
- cabling alteration

6. SELECTION PROCESS

Once the cost/benefit analysis has been accepted, the consultant is expected to prepare two sets of evaluation crite-
AUSTRALIAN MANUFACTURING PROBLEMS

Markets
Regionalised markets
Excessive product variety
Seasonal demand
Most markets over-supplied
Dumping from Europe/North America/Asia
Excessive product distribution costs
Reduced product lead-times
Improved quality requirements

Manufacturing
Powerful trade unions
High wage structure
Relatively low volumes
Limited choice of local vendors
Manufacture to licence rather than own design
Shortage of engineers, technicians and tradesmen
95% of CIM technology imported
Poor delivery performance
High inventories

Savings
Providing that the manufacturing systems are properly implemented and there is a strong management involvement, results can be produced similar to the following:
1. A typical reduction in inventory of 20 to 35 per cent.
2. Less risk of inventory shrinkage.
3. Improved cash flow.
5. Improved productivity — 5 to 10 per cent in fabrication areas in manufacturing, 25 to 40 per cent in assembly areas.
6. Overtime reduced by from 50 to 90 per cent.
7. Reduction of purchased material costs averaging 5 per cent annually.
8. Better customer service: typically improved into the 90 plus per cent range.
9. Reduce indirect staff by at least 10%.
10. Reduce lead-times by significant levels.

Major Software Features Criteria

<table>
<thead>
<tr>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
</tr>
<tr>
<td>Med</td>
</tr>
<tr>
<td>Low</td>
</tr>
<tr>
<td>1. Order Entry Capability to include insertion of both Data-Base and Non-Data-Base Items</td>
</tr>
<tr>
<td>2. Bill-of-Material capability to include both proportional bills and &quot;Same-As-Except&quot; features</td>
</tr>
<tr>
<td>3. Routings capability must have a &quot;Same-As-Except&quot; feature</td>
</tr>
<tr>
<td>4. Shop Floor Module should be able to load works orders directly into module</td>
</tr>
<tr>
<td>5. Shop Floor Module must have a Work-In-Progress monitoring capability</td>
</tr>
<tr>
<td>6. Ability to schedule work orders according to hierarchical priorities</td>
</tr>
<tr>
<td>7. Inventory module must be able to create re-order demands independently from MRP</td>
</tr>
<tr>
<td>8. Costing Module must include both a standard costing and a job costing feature. The former for Inventory Issuing and Receiving</td>
</tr>
<tr>
<td>9. Wide Ranging Purchasing Module</td>
</tr>
<tr>
<td>10. Software to have a Simple MRP module</td>
</tr>
<tr>
<td>11. The package must be an unsophisticated one</td>
</tr>
</tbody>
</table>

Vendor Ranking Criteria

<table>
<thead>
<tr>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
</tr>
<tr>
<td>Med</td>
</tr>
<tr>
<td>Low</td>
</tr>
<tr>
<td>1. Major Software Criteria Ranking</td>
</tr>
<tr>
<td>2. Maturity of the package and satisfactory performance in Australian sites</td>
</tr>
<tr>
<td>3. Application software support in Melbourne/Sydney</td>
</tr>
<tr>
<td>4. Training capability in Melbourne/Sydney</td>
</tr>
<tr>
<td>5. Financial stability of the vendors</td>
</tr>
<tr>
<td>6. Quality of Documentation</td>
</tr>
<tr>
<td>7. Price competitiveness of tender</td>
</tr>
<tr>
<td>8. Application software vendor user group</td>
</tr>
<tr>
<td>9. Software vendor to have established implementation procedures</td>
</tr>
<tr>
<td>10. Application software development team policy and procedures</td>
</tr>
</tbody>
</table>
Consultant Assisted Purchase of System

vendor's attitude as shown by his submission and his approach to the evaluation. Is the vendor serious about winning the business? Are demonstrations well conducted, and do the local representatives understand the product in sufficient detail? Does the vendor respond to difficult questions by saying that the feature is in the next release? This response is a giveaway. Beware! Speak to the product user group, if one exists, to find out how the vendor handles support.

The selection team and the consultant need to be aware that a software package selection is an intensely political activity. The team needs to flush out all shades of opinion and to achieve some consensus. Company executives often take extreme positions during a selection process which can lead to a shedding of company blood. Sometimes the consultant may be asked to arbitrate the situation.

7. CONCLUSION
The consultant has a definite role to play in the analysis, requirements definition and selection phases of the acquisition of manufacturing software. He should be expected to carry out the roles suggested in the paper conscientiously, producing reports and reporting back to the company's management for making key decisions about requirements and selection. He should be expected to document his recommendations thoroughly and provide alternative solutions where appropriate.

On the other hand, it is the company management's responsibility to set the terms of reference for the consultant clearly at the outset, and to provide the consultant with information, advice and guidance as required throughout the course of his investigation. The consultant especially requires guidance as to the company's objectives and priorities to ensure that the system that is finally specified is the one that suits the company and meets operational and fiscal targets, rather than the system that the consultant has proposed before to other clients, or the one that he favours in the absence of other constraints.

It is particularly important that the consultant is made aware of the company’s attitude to the incorporation of the MRP system into a wider manufacturing, or CIM, system and into the company's overall information systems. This will allow the consultant to assess properly the position of the system relative to new technology which might be applicable to the company.

8. REFERENCES
Guidelines for Privacy and Security in Computer Systems

W.J. Caelli†

With the maturing of the computer industry in the 1980s, the computer professional must now assume that computer and information systems are no longer used or even developed in a secure and safe environment. Into the 1990s the computer professional will be compelled to increasingly consider the privacy, integrity and authenticity problems associated with systems users, workstations, data networks and data bases. These concerns will not only be forthcoming from within the professional associations of computer people themselves but increasingly from society at large, the law, governmental organisations and standards setting bodies. This will impose new codes for professional practice in the overall area of security.

THE GROWING THREAT
The growing threat to computer systems from sources both internal and external to an organisation is rapidly being recognised by computer professionals. This represents a considerable change in the threat assessment over the last thirty years of general computer usage, particularly in the commercial area. The security threat to computer systems has accelerated in the 1980s because of:
— growing literacy in computer technology at the general society level,
— the general availability of low cost computer technology at the consumer level,
— widespread usage and access to computer systems, and
— total dependence, in many cases, of both public and private enterprises on information systems leading to the problem of a “denial of service” threat to the organisation’s actual operation. (In this sense the computer system provides a service that cannot be provided manually or that has no manual back-up.)

These threats to the uninterrupted and continuing “smooth functioning” of an information system may be assessed in terms of the likelihood of attack on the system from both within an organisation and from sources external to the organisation (the so-called “hacker” problem, for example). The users of an information system are now looking to computer professionals to create information systems that can be increasingly “trusted” to provide correct and timely services on request and that function according to specification.

RESPONSIBILITY IN AN OPEN ENVIRONMENT
The computer professional is now faced with the overall responsibility for the design, development, implementation, maintenance and enhancement of information systems that must be assumed to operate in a “hostile” and “open” environment. This environment is characterised by:
— varying levels of security requirements,
— application systems of varying security requirements running concurrently,
— users who can no longer be “trusted” to “behave” (in terms of terminal or workstation usage) according to accepted levels of responsibility,
— application software in the information system itself that similarly itself cannot be “trusted”, i.e. perform the functions as specified and no more, and to correctly operate in conjunction with other software systems and to be isolated from them,
— the presence of software that may not have been authorised for actual operation in the information system,
— interconnection of computer systems into networks whereby the reliability of some connected systems and their data and program bases may be unknown, and
— the presence of “illicit” users must be assumed from both inside and outside the organisation.

The computer professional is now faced with the implementation of security features in software and hardware systems in line with growing international standards, such as the International Standards Organisation’s (ISO) Open Systems Interconnection (OSI) model and its security extensions. Overall network and system security may become integral parts of the information system design at the “application layer” within this model. It cannot be assumed that security functions will be the total responsibility of lower layers of the OSI model, or of systems software or hardware sub-systems.

PRIVACY, INTEGRITY AND AUTHENTICITY
The computer professional now recognises that both data and programs are subject to clear security provisions. Both have privacy, integrity and authenticity requirements, particularly from the users viewpoint. These requirements, however, may differ radically between data and programs and separate security statements must be set out for each.

Copyright © 1987, Australian Computer Society Inc.

† Eracom Pty Ltd, 26 Greg Chappell Drive, Burleigh Heads, Queensland, 4220, Australia. Invited paper.
For example, an algorithm embedded in a program may be the proprietary property of an enterprise and be more valuable or subject to disclosure constraints than the data on which it operates.

Thus the computer professional will be increasingly involved in the development of information, systems and risk analysis techniques and methodologies that fully reflect security requirements. These requirements may be those set by management, the law, government or society in general. Privacy and integrity constraints need to be analysed in terms of “subjects” and their access to “objects” within the overall computer system. Such “subjects” in a computer system may include computing processes, network and terminal access, identifiable individuals and so on. Computing “objects” may include again computing processes as well as data structures and actual data instances. For example data identifiable to an actual individual may have certain specific security requirements, as is the case with the United Kingdom’s “Data Protection Act”. However, in the future, such security restrictions may apply to the safe storage and processing of any identifiable data, e.g. data associated with an actual company or enterprise. Such data may also include the “constraints” sets that form part of the set of rules set up to create and maintain an integrated data base.

SYSTEMS EVALUATION
Professional responsibility now includes the examination and evaluation of the security facilities of any computer system and data network structure which is to be used in the creation of an information system. This normally involves assessment of the hardware, operating system, communications interfaces and like sub-systems and the analysis and definition of any additional security facilities that may be required to meet the stated risk assessment for the information system. Such an assessment may include additional security sub-systems that must be included at the application development time and/or modifications or enhancements to the security features at the systems software and hardware levels. For example, a simple password entry facility at “LOGON” time by a terminal may not be sufficient to meet the requirements for access control and auditability for the protection of proprietary and confidential data bases and program collections. Additional terminal and individual identification schemes may need to be employed such as possession of machine readable or self-contained “tokens”, usage of biometric devices such as fingerprint scanners, and so on.

Moreover, the risk assessment may point to the need to incorporate such access control devices into computer systems design at a number of points. User authentication may be required not only at the time of original terminal access to the information system but at the time that individual programs are started or data bases are accessed at various levels within the system.

Analysis of the data network will incorporate privacy, integrity and authentication requirements based upon:
— terminal type, location and authentication procedures,
— user authentication,
— data transfer privacy including, if necessary, privacy of data flow statistics and usage,
— data transfer/transaction integrity required over and above any integrity technique supplied within the network technology,
— transaction authenticity, and
— network audit needs.

Terminal/workstation installation and usage may present particular security problems in a number of ways. The physical environment, location and accessibility of the terminal, in contrast to mainframe, LAN server or host systems, may not be controllable and may even be “hostile”. The reliability of the terminal may need to be considered against expected usage and predicted normal/error behaviour in the data network. Where the terminal unit is a personal computer consideration may need to be given to the incorporation of specific access control sub-systems and protection schemes for data/program storage media, such as “floppy” diskettes, tape cartridges and so on.

RISK ANALYSIS AND ASSESSMENT
The usage of formal risk analysis and assessment methodologies will increasingly become part of the responsibility of the computer professional as well as the computer security specialist and the EDP auditor. The responsibility of notification to management of the necessity to perform such analysis should also lie with the computer professional. Management may also need to be advised about its participation in the risk analysis function and particularly in the evaluation and assessment of the analysis and its interpretation into specific directives. This management role also applies to the preparation and continuing review, at regular intervals, of the formal risk assessment document.

The responsibility of the computer professional will increasingly be to recommend cost effective solutions to the security requirements set out by the risk assessment study, including the selection of appropriate hardware and software sub-systems and the establishment of management/personnel procedures. Encryption and access control equipment may need to be incorporated into sensitive information systems and the impact of such sub-systems on the performance and human acceptability of the system must be assessed. In addition security systems such as encryption devices and procedures may involve additional managerial procedures and technical requirements. For encryption equipment this may include key management schemes for the efficient and secure operation of the equipment at the terminal, data network and host computer systems levels.

LEGAL MATTERS
There is no doubt that the computer professional will be increasingly affected by government legislation related to the use of computer systems and to the storage of data within them. In addition problems of copyright in relation to computer programs and data structures may also apply. Thus the computer professional must seek the written
definition by management and/or seek written legal advice of any legal liabilities which the professional may have in the creation of an information system. These liabilities may include those associated with accidental or deliberate exposure of identifiable individual data or proprietary programs.

CONCLUSIONS
This paper has set out some aspects of information system security that are or will soon be of concern to the computer professional. The widespread availability of computer technology and the increased knowledge of that technology by the general public mean that the risks posed to information systems will increase as their penetration into all aspects of commercial life, in particular, increases. Moreover, the complete dependence on these systems for the provision of necessary services means that deprivation of the computer services may have catastrophic effects on a number of essential services. The computer professional is thus faced with the development, installation and maintenance of information systems that may be deemed to exist in a "hostile" environment, being attacked from within an organisation as well as from outside it. This new environment will lead to a set of professional practices in relation to the security of information systems and to the maintenance of the privacy and integrity of data and program bases held within them.

REFERENCES
A Checklist of Factors to be Considered when Negotiating Software Agreements

Norbert F. Riedl†

In this article guidance is offered to potential software purchasers concerning the nature of agreements that should be made with vendors. Although the author is not legally trained, many of the recommendations flow from discussions with professional legal persons and from negotiations with vendors. Although the checklist and guidelines were developed for academic institutions, slight modification should render them appropriate to your commercial or government group.

Keywords and Phrases: Software licence, software agreement, software purchase contract.

1. INTRODUCTION

Most, if not all, legally acquired software is accompanied by a software agreement (license). These agreements vary in their length and complexity and are often difficult to interpret in a manner that allows the lay person to get a clear picture of what is being agreed. In many software agreements there are conditions which the purchaser (or licensee) will not particularly like but will find that the software supplier will not be prepared to waive or alter. Many of these are “industry standard” conditions. There may be some which we believe are totally unacceptable and must not be agreed to. (Even major vendors have been willing to alter their “standard” terms and conditions to get sales!)

If you have any doubts at all do not hesitate to get advice from either your Purchasing Officer or seek legal advice.

Section 2 presents a checklist of factors to be considered. Section 3 presents explanatory notes which elaborate on some of the questions in the checklist and give examples of some “standard” clauses which are either unacceptable or at least should be seriously challenged.

2. SOFTWARE AGREEMENT CHECKLIST

The checklists shown below are intended as a guide to help purchasers of computer software in checking the conditions of an agreement under which the software is being acquired. The list attempts to point out some of the more important conditions which should be clearly stated in any software agreement.

3. EXPLANATION OF CHECKLIST

The reference numbers refer to those in the checklist.

Ref: 1.1, 1.2 — Vendors Rights to Sell/Licence

If the software vendor is not the actual owner (e.g. copyright or patent holder) it is appropriate for you to ascertain his relationship and licencing agreement with the owners. If doubtful, ask to see his accreditation to sell or license the software on behalf of the owners.

Ref: 1.3

It is unlikely that you will actually be purchasing software. So “ownership” agreements are rare except where an individual is selling a package for you to market (this does occur in the “hobby” software market).

Ref: 1.4 — Changes in Configuration

Most suppliers of software for mainframe and mini computers allow some changes to the computer configuration without these affecting the licencing agreement. However, some restriction may be imposed as can be seen by the following example:

“Any change in the manufacture model or location of the computer on which the Software is used by the Licensee and any change in the details of an Additional Site which is included in this Agreement and details of any other Additional Sites which are to be included in this Agreement must be communicated to XYZ Ltd within six (6) weeks of change occurring.”

Where some form of networking of several CPUs is involved, the effects of changes to the network configuration should be clearly stated and agreed to by all parties involved. If no mention is made of any effect on the software licence by possible configuration changes, make certain that attaching more, or different peripherals, does not invalidate your agreement.

It should also be noted that some suppliers impose
## Checklist of Factors

### VENDORS RIGHT TO SELL/LICENCE

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
<th>Not Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Is the vendor the owner of the product?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IF NOT:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Can the vendor show you the agreement that exists between the Vendor and the owner?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— What type of agreement does the vendor have?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Agency?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— OEM? (Original Equipment Manufacturer)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Other?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2 Is the licence agreement to be signed, an agreement between your Institution and:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— The vendor?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— A distributor?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— The owner?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3 If “purchasing” the software, what type of agreement are you entering into:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Ownership of software?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— User licence with:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Right to use?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Right to copy?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4 The “purchase price” is:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— A once only fee allowing you to do what you like with the software?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— A once only “right to use” fee with a “right to copy”?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— An annual “right to use” fee?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

restrictions on how the software is being configured, e.g.: “The Licensee agrees to limit use of the Software to use as a whole. No part of the Software may be used separately from the whole. XYZ Ltd shall be entitled to call for an annual statement signed by a responsible officer of the Licensee that the provisions of this Clause have been fully complied with.”

Such a condition can be a nuisance and may seriously affect the way an organisation conducts its affairs.

Restrictions on who can use the software and the purpose for which it may be used are varied and must be checked thoroughly. In many institutions, organisational affiliations are not always clearly defined (e.g. joint research ventures with other organisations and subsidiary companies). A condition such as:

“The Licensee agrees to limit access to the Software to members of the Licensee’s organisation and to members of the Additional organisations specified herein.”

could cause difficulties where the site is used by persons from many organisations (as is the case at Computer Centres).

One well known supplier of micro computer software specifies that:

“The products provided by XYZ Ltd are intended for commercial use only.”

Does this affect the use of the products at an Educational (i.e. non-commercial) site?

Some software suppliers restrict the discounted licenses negotiated with an organisation to educational use only. In some of these cases, the software cannot be used for any administrative functions, without additional charges.

In other instances, “educational” does not include “commercial education” i.e. where tuition fees are paid to a “profit making” organisation (even if it is a subsidiary company). In addition, it may not cover the situation where users are charged for use of the software on your computer, e.g.:

“If the Primary Site or any Additional Site included in this Agreement sells computer services either as a main function or as a subsidiary activity, the Licensee agrees to pay, quarterly in arrears, XYZ Ltd a royalty of 25% of the cost of all work invoiced to third parties for the use of the Software at the Primary Site or from any Additional Site which is included in this Agreement. XYZ Ltd shall be entitled to call for an annual statement signed by a responsible officer of the Licensee that the...
# Checklist of Factors

## CHECKLIST

### 2. RIGHT TO USE

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
<th>Certain</th>
</tr>
</thead>
</table>

2.1 If the proposed licence agreement is for a "right to use" licence, check the following features:

- Is the licence for one computer only?
  - If so, are there any clauses which would affect the agreement if the configuration is altered in any way? (___ ___ ___)
  - Does the licence apply to any number of users of the computer (i.e. multiple access stations)? (___ ___ ___)
- Are there any restrictions on the use of the computer and the software on that computer, e.g.
  - Restricted to "educational purposes?" (___ ___ ___)
  - Restricted to "teaching purposes?" (___ ___ ___)
  - Restricted by use by the purchaser's main organisation (i.e. excluding subsidiary companies)? (___ ___ ___)
  - Restricted to users of the software who do not pay a fee for access to the computer? (___ ___ ___)
  - Other restrictions? (___ ___ ___)

- Is the licence affected if the computer is networked with other computers? (___ ___ ___)
  - If so, how? (___ ___ ___)

2.2 If the licence is for multiple computers, the same general conditions as for a single computer licensing agreement should apply. In addition, are the specific machines to which it applies clearly identified?

- To which and how many computers does the licence apply? (___ ___ ___)

The following should also be checked:

- Are the specific machines to which it applies clearly identified? (___ ___ ___)

---

Provisions of this clause have been fully complied with.

**Ref: 2.2**

Licences for multiple computers vary from supplier to supplier. On larger computers it is normal to pay for one "right-to-use" licence plus a number of "right-to-copy" licences for subsequent CPUs. The latter are typically 10%-30% of the cost of a "right-to-use" licence. With microcomputers the situation is more varied and must be checked carefully.

Many suppliers insist on a separate licence for each microcomputer even if they are networked and/or share common peripherals. Other suppliers give multiple-user discounts. The following is an extract from a software agreement from a major international supplier:

"MULTI-PROCESSOR USE AND MULTI-USER LICENCE FEES.

In the event that End User's computer is of multi-processor design whereby multiple users may access common disk memory and therefore software programming the following user fees shall apply:

- 1 user .......................... 1 licence required
- 2-5 users .......................... 2 licences required
- 6-24 users .......................... 3 licences required
- 25 or more users ........ 1 additional licence required for each 20 users or fraction part thereof.

End User shall have the enclosed form "Multi-Processor Design System Configuration" Exhibit C completed upon initial purchase of XYZ software. Additional "Multi-Processor Design System Configuration" forms shall be provided by End User to the XYZ"
## Checklist of Factors

### 3. Right to Copy

If a “right to copy” licence is proposed, the same questions as for a “right to use” licence should be answered.

In addition, the following should be checked for both the “right to use” and “right to copy” licences:

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
<th>Not Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Does the proposed licencing agreement clearly state that you may make copies of the software (without additional charge or liability for copyright infringement) for:</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Archiving</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— To replace defective copies.</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— For program error verification.</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— For system security (backup).</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— If it does allow copying for the above purposes, are the necessary utilities provided under the agreement?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— If it does not allow all of the above:</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>WHY NOT? THEY ARE ESSENTIAL!!</strong></td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>3.2 Does the proposed licencing agreement clearly allow the copying of all relevant documentation:</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Only as allowed under the Australian Copyright Act?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Only for educational purposes?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Other specified purpose(s).</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Not at all?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>3.3 If copying for education/teaching purposes is expressly permitted:</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Are there additional charges?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Can it be onto any medium (e.g. paper, magnetic tape, microfiche etc.)?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>3.4 Ensure you know the charging structure.</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Is the charge for the licence clearly stated?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Does the charge apply to each copy made?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— Are the target computers clearly identified?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>— If so, how?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>3.5 Are the copies made covered by the normal warranty period specified in the agreement?</td>
<td>-----</td>
<td>----</td>
<td>-------------</td>
</tr>
</tbody>
</table>

Dealer or Distributor from whom the original license was purchased or if that is not possible to XYZ itself when the numbers of users on a multi-processor design system previously reported shall change. End User shall comply with the above fee schedule and multiple use reporting system for Design System Configuration supporting Multiple Users on a Single Processor.”

**Note:** This applies not only to multiple CPUs but also to multiple users and it is the licensee’s responsibility to notify the licensor if the number of users changes.

Most suppliers merely state that a licence is need for each system e.g.:

**“Licence**

Each program license granted under this Agreement authorises the Customer to use the Licensed Program in any machine readable form on any single computer system (referred to as System). A separate license is required for each licensed System on which the
Licensed Program will be used.
This Agreement and any of the licensed programs or materials to which it applies may not be assigned, sublicensed or otherwise transferred by the Customer without prior written consent from XYZ Ltd. No right to print or copy in whole or in part the Licensed Programs is granted except as hereinafter expressly provided."

Ref: 3 — Right to Copy
Ref: 3.1
Most reputable software suppliers allow the copying of software for backup and/or archival purposes. Some stipulate that copies can only be used on the machine for which the original software was purchased. This condition is generally acceptable for micros, but must be queried if you wish to use a second computer as a back-up machine. Some suppliers also stipulate the numbers of copies which you can have at any one time. Again this can be important if you have a son-father-grandfather cycle of back-up for your complete software repertoire.

There are suppliers of software, mainly for microcomputers, where you are not able to copy the software at all. In these cases, if you have a malfunction and need another copy you must get it from the supplier. Invariably there will be a service fee for this and in extreme cases you will be charged another licence fee. Even if there is no charge, the time delay in getting a new copy can be annoying.

The following are extracts from contracts which illustrate some of the above restrictions:

a. "ARTICLE III. ARCHIVAL COPIES. End User may make archival copies of those portions of XYZ Ltd's product(s) that are provided on machine-readable media provided such copies are for the End User's personal use on one microcomputer and that no more than one such copy is in use at any time unless End User has paid for multiple copy use as described in ARTICLE IV MULTIPLE COPY USE of this Agreement.

ARTICLE IV. MULTIPLE COPY USE. XYZ Ltd user licenses are applicable to a single microcomputer installation. In the event End User intends to use an XYZ Ltd product or any part thereof on more than one microcomputer, the license fee for each such multiple use must be purchased. Quantity discounts in accordance with XYZ Ltd policy will apply in the event of simultaneous use. A license must be obtained for each instance of possible simultaneous execution by a separate microprocessor."

b. "Permission to copy or modify Licensed Programs. The Customer shall NOT copy, in whole or in part, any Licensed Documentation which is provided by XYZ Ltd in printed form under this Agreement. Additional copies of printed materials may be acquired from XYZ Ltd or its authorised distributors. Any Licenced Programs which are provided by XYZ Ltd in machine readable form may be copied, in whole or in part, in sufficient number for the use by the Customer with the Licensed System to understand the contents of such machine readable material to modify the Licensed Program as provided herein for back-up purposes or for archive purposes provided however that no more than five (5) copies will be in existence under any License or any one time without prior written consent from XYZ Ltd. The Customer agrees to maintain appropriate records of the number and location of all such copies of the Licensed Programs."

Both of the above are acceptable but do have restrictions which should be noted.

Ref: 3.2, 3.3
The copying of documentation is usually not allowed unless specifically negotiated. Most suppliers of software for main frames and mini computers are prepared to negotiate conditions for copying documentation for non-profit-making educational purposes. This is however seldom the case with suppliers of software for microcomputers. In most cases additional copies of the documentation must be purchased (although some will negotiate bulk discounts).

Extreme cases have been encountered where, to get multiple copies of the documentation, it is necessary to get a complete multiple license agreement.

Ref: 4 — Warranty
Very few software suppliers give any meaningful warranties for their products. It is very much a "buyer beware" situation. The following is a representative example of the lack of warranty for software:

"Disclaimer of Warranty
XYZ makes no warranties with respect to the Licensed Programs.

Limitation of Liability
The foregoing warranty is in lieu of all other warranties expressed or implied including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event will XYZ be liable for consequential damages even if XYZ has been advised of the possibility of such damages."

This clause is totally unacceptable as it provides no condition to rectify a serious deficiency in the product. Some suppliers will at least allow you to get your money back under certain conditions, e.g.:

"Every care has been taken to develop and test the Software for accuracy and XYZ Ltd believes in good faith that the Software will perform satisfactorily if properly installed on a suitable computer. However, no warranty is expressed or implied by this Agreement as to the accuracy and performance of the Software. The Licensee accepts that this Agreement does not impose any liability on XYZ Ltd for any failure or defect in the Software or for any loss or injury however arising in connection with the use of the Software by the Licensee. If the Licensee demonstrates to the reasonable satisfaction of XYZ Ltd that as a result of fault in design, material or
The
AUSTRALIAN COMPUTER JOURNAL
VOLUME 19
February 1987 to November 1987

CONTENTS

Contributions

1. February, 1987
O'Neil, J.L.

McGrath, G.M.

Tse, T.H.

Belbin, L.

Plausible Reasoning
The Transition to Fifth Generation Technology: Conceptual Schema
Implementation
Integrating the Structured Analysis and Design Models: a Category-Theoretic Approach
The Use of Non-hierarchical Allocation Methods for Clustering Large Sets of Data

2. May, 1987
Galliers, R.D.
Fry, M.R.
Patrick, J., Poole, P.C. and Lions, J.
Brent, R.P.
Leung, C.M.R., and Nijssen, G.M.

Information Systems Planning: A Manifesto for Australian-Based Research
The Transport Layer in a Microcomputer Network
Selected Papers from the Tenth Australian Computer Science Conference: Guest Editors' Introduction
A Linear Algorithm for Data Compression
From a NIAM Conceptual Schema into the Optimal SQL Relational Database Schema
A Graphics Oriented Deductive Planning System
Some Experiments in Decision Tree Induction
Reproducible Tests in CSP
Specifying the Static Semantics of Block Structure Languages

3. August, 1987
O'Callaghan, J.F.
Thornton, B.S., Todter, C., Botten, L. and Caden, M.
Fraser, D.
Newman, B.B. and Hildebrandt, J.
Benke, K.K. and Skinner, D.R.
Shipman, A.L.

Special Issue on Digital Image Processing: Guest Editor's Introduction
Moiré Fringe Methods in Aerofoil Sail Shape Measurement for the "Kookabura" 12 Metre Yachts
A Conceptual Image Intensity Surface and the Sampling Theorem
Blind Image Restoration
Segmentation of Visually Similar Textures by Convolution Filtering
Implementing Relaxation Labelling with Neural Networks

Continued overleaf...
Craig, M.D. and Green, A.A.  
Registration of Distorted Images from Airborne Scanners 148-153

Nichol, D.G., Fiebig, M.J., Whatmough, R.J. and Whitebread, P.J.  
Some Image Processing Aspects of a Military Geographic Information System 154-160

Robinson, D.E. and Knight, P.C.  
Image Processing in Diagnostic Ultrasound 161-166

Sale, A.H.J.  
Optimization Across Module Boundaries 167-173


Lions, J.  
The Australian Computer Journal: Twenty Years On 185-189

Montgomery, A.Y.  
The Australian Computer Journal: A Twentieth Anniversary Editorial - Twenty Deadly Sins Against Professionalism and Progress in Australian Computing 190-195

Ferme, W.D.  
What Could a Client Reasonably Expect a Consultant to do When He is Engaged to Assist in the Purchase of a Manufacturing System 196-202

Caelli, W.J.  
Guidelines for Privacy and Security in Computer Systems 203-205

Reidl, N.F.  
A Checklist of Factors to be Considered when Negotiating Software Agreements 206-214

Coldwell, R.A.  
Non-professional Practices in Computing: Some Thoughts on the Next Decade or So 215-218

Farrell, R.  
A Guide to Software Quality for the Developer and Unwary User 219-221

Clarke, R.  
The Computing Professional’s Bookshelf 222-223

Woodings, T.  
The Professional Development and Continuing Education of Computing Practitioners 224-230

Miscellaneous

Editorial 1

Helpers Needed 15

Call for Papers for Conferences 24, 41, 68, 139, 184

Call for Papers for Special Issues 31, 125

Book Reviews 42-6, 105-111, 174-183

Letter to Editor and Reply 47-8

Appointment of a New Editor 68

The Journal’s Referees 112

Continued overleaf ...
**TITLE INDEX**

*(Titles in italics are for book reviews.)*

<table>
<thead>
<tr>
<th>Title</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>68000 Micocomputer Systems: Designing and Troubleshooting</td>
<td>235-6</td>
</tr>
<tr>
<td>A Checklist of Factors to be Considered when Negotiating Software Agreements</td>
<td>206-214</td>
</tr>
<tr>
<td>A Conceptual Image Intensity Surface and the Sampling Theorem</td>
<td>119-125</td>
</tr>
<tr>
<td>A Genstat Primer</td>
<td>182</td>
</tr>
<tr>
<td>A Graphics Oriented Deductive Planning System</td>
<td>76-83</td>
</tr>
<tr>
<td>A Guide to Software Quality for the Developer and Unwary User</td>
<td>219-221</td>
</tr>
<tr>
<td>A Linear Algorithm for Data Compression</td>
<td>64-68</td>
</tr>
<tr>
<td>A Quarter Century of IFIP</td>
<td>174</td>
</tr>
<tr>
<td>Ada in Practice</td>
<td>42</td>
</tr>
<tr>
<td>Ada: managing the transition</td>
<td>181</td>
</tr>
<tr>
<td>Advances in Artificial Intelligence</td>
<td>43</td>
</tr>
<tr>
<td>Algorithms, their complexity and efficiency</td>
<td>180</td>
</tr>
<tr>
<td>Algorithms: the Construction, Proof and Analysis of Programs</td>
<td>43</td>
</tr>
<tr>
<td>An Introduction to Computer-Aided Design and Manufacture</td>
<td>44</td>
</tr>
<tr>
<td>An Introduction to Data Types</td>
<td>231</td>
</tr>
<tr>
<td>Blind Image Restoration</td>
<td>126-133</td>
</tr>
<tr>
<td>C For Programmers</td>
<td>181</td>
</tr>
<tr>
<td>Cellular Logic Image Processing</td>
<td>234-5</td>
</tr>
<tr>
<td>Choosing and Using CMOS</td>
<td>236</td>
</tr>
<tr>
<td>Cobol 85 Reference Summary</td>
<td>231</td>
</tr>
<tr>
<td>Computer Assisted Decision Making</td>
<td>45</td>
</tr>
<tr>
<td>Computer Hardware Description Languages and their Applications</td>
<td>106</td>
</tr>
<tr>
<td>Computer Security: the practical issues</td>
<td>105</td>
</tr>
<tr>
<td>Computer Users’ Data Book</td>
<td>111</td>
</tr>
<tr>
<td>Computing '85: A Broad Perspective of Current Developments</td>
<td>174</td>
</tr>
<tr>
<td>Concurrency and Programming Languages</td>
<td>232</td>
</tr>
<tr>
<td>Concurrent Programming in Ada</td>
<td>46</td>
</tr>
<tr>
<td>CPIM 80 System Programming</td>
<td>45-6</td>
</tr>
<tr>
<td>Data Base Organization for Data Management</td>
<td>183</td>
</tr>
<tr>
<td>Desktop Publishing: The Book</td>
<td>111</td>
</tr>
<tr>
<td>Dynamical Systems and Cellular Automata</td>
<td>234-5</td>
</tr>
<tr>
<td>Formal Aspects of VLSI Design</td>
<td>183</td>
</tr>
<tr>
<td>Framework II for Writers</td>
<td>46</td>
</tr>
<tr>
<td>From a NIAM Conceptual Schema into the Optimal SQL Relational Database Schema</td>
<td>69-75</td>
</tr>
<tr>
<td>Fuzzy Mathematical Techniques with Applications</td>
<td>109</td>
</tr>
<tr>
<td>Getting Started with WORDSTAR + Mailmerge + Spellstar</td>
<td>178</td>
</tr>
<tr>
<td>Guidelines for Privacy and Security in Computer Systems</td>
<td>203-205</td>
</tr>
<tr>
<td>High Tech: Window to the Future</td>
<td>111</td>
</tr>
<tr>
<td>Image Processing in Diagnostic Ultrasound</td>
<td>161-166</td>
</tr>
<tr>
<td>Implementing Relaxation Labelling with Neural Networks</td>
<td>140-147</td>
</tr>
<tr>
<td>IMS (DL1) Data Base Organisation and Performance</td>
<td>231</td>
</tr>
<tr>
<td>Information Systems Design Methodologies: Improving the Practice</td>
<td>174-5</td>
</tr>
<tr>
<td>Information Systems Planning: A Manifesto for Australian-Based Research</td>
<td>49-55</td>
</tr>
<tr>
<td>Integrating the Structured Analysis and Design</td>
<td>25-31</td>
</tr>
<tr>
<td>Integrity Analysis: A Methodology for EDP Audit and Data Quality Assurance</td>
<td>42</td>
</tr>
<tr>
<td>Intelligent Machinery -- Theory and Practice</td>
<td>108</td>
</tr>
<tr>
<td>Interface Fundamentals in Microprocessor-Controlled Systems</td>
<td>107</td>
</tr>
<tr>
<td>Introduction to Computer Systems</td>
<td>181-2</td>
</tr>
<tr>
<td>Introduction to Finite Fields and their Applications</td>
<td>182-3</td>
</tr>
<tr>
<td>Introduction to the Graphical Kernel Systems (GKS)</td>
<td>233</td>
</tr>
<tr>
<td>Language and Artificial Intelligence</td>
<td>180</td>
</tr>
<tr>
<td>Microcomputer Operation, Troubleshooting, and Repair</td>
<td>107</td>
</tr>
<tr>
<td>Modelling Techniques and Tools for Performance Analysis '85</td>
<td>232-3</td>
</tr>
<tr>
<td>Modula-2 Discipline &amp; Design</td>
<td>110</td>
</tr>
<tr>
<td>Modula-2: Constructive Program Development</td>
<td>110</td>
</tr>
<tr>
<td>Moire Fringe Methods in Aerofoil Sail Shape</td>
<td>115-118</td>
</tr>
<tr>
<td>Measurement for the &quot;Kookabura&quot; 12 Metre Yachts</td>
<td></td>
</tr>
<tr>
<td>Networks in Office Automation</td>
<td>46</td>
</tr>
<tr>
<td>Non-professional Practices in Computing: Some Thoughts on the Next Decade or So</td>
<td>215-218</td>
</tr>
<tr>
<td>Numerical Recipes</td>
<td>176-7</td>
</tr>
<tr>
<td>Object Oriented Programming: An Evolutionary Approach</td>
<td>43-4</td>
</tr>
<tr>
<td>Optimization Across Module Boundaries</td>
<td>167-173</td>
</tr>
<tr>
<td>Plausible Reasoning</td>
<td>2-15</td>
</tr>
<tr>
<td>Practical Data Communications: Modems, Networks and Protocols</td>
<td>107-8</td>
</tr>
<tr>
<td>Practical Program Development Using JSP</td>
<td>175</td>
</tr>
<tr>
<td>Proceedings of the Fifth British National Conference on Databases</td>
<td>175-6</td>
</tr>
<tr>
<td>Program Evolution</td>
<td>179-80</td>
</tr>
<tr>
<td>Programming the Macintosh - An Advanced Guide</td>
<td>177-8</td>
</tr>
<tr>
<td>Prolog: A Relational Language and Its Applications</td>
<td>231-2</td>
</tr>
</tbody>
</table>

*Continued overleaf ...*
**CHECKLIST**

### 4. WARRANTY

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
<th>Not Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Are warranty conditions clearly stated?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is the warranty conditional on taking out a maintenance agreement?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Are the conditions under which the warranty does <em>not apply</em> or <em>is no longer</em> valid, clearly stated?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does the warranty clearly state:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Start of warranty?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Duration of warranty?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— All items, products etc. covered by this warranty?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Are any items not covered?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— If not, why?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2 What are the vendor's obligations under the warranty?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Are all parts, materials and labour costs covered?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Is it on a repair/correction basis?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Is it on a replacement basis?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— If repairs/replacement is not on the purchaser's premise are there any freight charges?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Is the decision to repair or replace entirely at the vendor's discretion?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3 Are any performance guarantees given with the software?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Workmanship, the Software has substantially failed to operate on the computer specified in the Order Schedule the Licensee shall be entitled to a refund of the Licence Fee.

Except as provided below NO OTHER WARRANTY IS EXPRESSED OR IMPLIED AS TO THE QUALITY OR FITNESS OF XYZ PRODUCTS FOR ANY PARTICULAR PURPOSE OR OTHERWISE VENDOR SHALL NOT BE LIABLE FOR CONSEQUENTIAL LOSSES OR DAMAGES.

Where the Purchaser is a consumer (within the meaning of the Trade Practices Act) of goods or services supplied hereunder the conditions and warranties implied by that Act will apply to the extent to which the Contract relates to the supply of those goods or services. Purchaser acknowledges that Vendor is unaware of the consequences to the Purchaser of failure of the goods purchased. The Purchaser relies solely on his own skill and judgement in all respects and in particular to enable him to decide that the goods purchased are reasonably fit for the purpose for which they are being acquired and to decide to purchase those goods.

Other than the type of conditions represented above, software vendors generally give no performance guarantees. Thus, unless the licensee can prove that the software is actually defective he has almost no recourse e.g.:

"LIMITED WARRANTY POLICY. XYZ Ltd warrants that all materials furnished by XYZ Ltd constitutes an accurate manufacture of XYZ Ltd products and will replace any such XYZ Ltd furnished material to be thus defective provided such defect is found within ten days of purchase by End User. However, XYZ Ltd makes NO express or implied warranty of any kind with regard to performance or accuracy of data of any kind nor for any consequential damages resulting therefrom whether through loss or inaccuracy of data of any kind nor for any consequential damages resulting therefrom whether through XYZ Ltd negligence or will not honour any warranty for which there is no signed End User Agreement Acknowledgement on file at XYZ Ltd offices. The products provided by XYZ Ltd are intended for commercial use only."

It should also be noted that some software suppliers provide a warranty which is conditional on the licensee agreeing to enter into a software maintenance contract for a specified period (e.g. one year) following the warranty period.

An example of such a conditional warranty can be seen in the following statement:

"We warrant that the media (magnetic tape, flexible disc, etc) on which the software is delivered is of high
<table>
<thead>
<tr>
<th>CHECKLIST</th>
<th>SUPPORT/MAINTENANCE</th>
<th>Yes</th>
<th>No</th>
<th>Not Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Is the vendor the sole support agent for the software?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If not, who else can support it:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Original owner?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Other agents or OEMs?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Purchaser?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2 If the purchaser can support the software will the supplier provide access to and copies of:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Source code?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— All relevant technical literature?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— All relevant utilities to support the software?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Is a period specified for which the supplier guarantees supply of the above support tools?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Does own support affect subsequent software upgrades from the supplier?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Does own support affect the warranty?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— Are there charges for the provision of the support tools specified?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHECKLIST</th>
<th>ACCEPTANCE OF TERMS</th>
<th>Yes</th>
<th>No</th>
<th>Not Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Are the conditions for acceptance of the proposed terms clearly stated?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.2 When does the software become the purchaser’s responsibility:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— On delivery of the software medium?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(BEWARE!!)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— On installation of the software:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— By the vendor?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— By the purchaser?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— By an agent?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— On payment?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If so, do normal commercial payment terms apply?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— After a specified time limit?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>— After specified (and agreed upon) performance tests?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Quality, i.e. that can be loaded and executed. *XYZ Ltd does provide the initial 90 days of software support at no charge if the customer purchases some level of support for one year following the initial 90 days period.* XYZ’s hardware and software support services are designed to provide equivalent services during warranty as contracted for the long term under an XYZ Ltd Customer Support Services Agreement. The flexibility in this policy enables customers to receive the service most appropriate to their needs.”

Ref: 5 — Support/Maintenance
Irrespective of whether the original owner or an accredited agent provides the support, the quality of the support cannot be realistically ascertained in advance. Almost every supplier has different levels and conditions for the support available for their software products. If you are negotiating the facility to provide your own support, make certain that you get all necessary software tools and carefully check to what extent the supplier is willing and able to help should you encounter difficulties. “Own” support may
### Checklist of Factors

#### Checklist

##### 7. Indemnity

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
<th>Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Will the vendor indemnify your Institution against any claim for infringement of any copyright or patent made against your Institution when using products acquired from the vendor?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If so to what extent?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If the vendor gives an indemnity, is the vendor able to fully cover any agreed indemnity amount?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Checklist

##### 8. Disputes

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
<th>Certain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Will the vendor accept an independent arbitrator (nominated by the purchaser) to help in settlement of software related disputes?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What law (country or state) governs the contract?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

have serious ramifications if you wish to upgrade to a later version of the supplier’s software in future.

**Ref: 6 — Acceptance of Terms**

Normally an agreement comes into force when both parties have signed it. However, the conditions by which the software becomes your responsibility should be carefully checked before purchasing it. The best type of agreement is where the software is not your responsibility until some agreed upon trials or period has elapsed. This can normally be negotiated with major suppliers for software on mainframes and mini-computers.

*Unfortunately it is rarely the case with software for micros.* Where possible, insist upon at least “normal commercial payment terms” when acquiring software. This implies a normal period of about four weeks after receipt of invoice to when it is actually paid. If any problem arises in that period you at least still have the money to be used as leverage. Conditions which indicate that the software is no longer the vendor's responsibility after despatch are totally unacceptable, as is the condition such as “opening of the goods constitutes acceptance”. It is questionable whether such a statement has any validity in Australia.

**Ref: 7 — Indemnity**

If the vendor does not have an indemnity clause for actions against the user with respect to third party actions claiming infringement of copyright, patent or trademark, there is no advantage in asking for one. The following is an example of a fair statement:

> **PATENT, TRADEMARK OR COPYRIGHT INDEMNITY.**
>
> XYZ Ltd shall, except as otherwise provided below, defend or settle any claim made or any suit or proceeding brought against Buyer so far as it is based on an allegation that any product furnished hereunder infringes a patent, trademark or copyright of the country in which Buyer takes delivery of said product, if notified promptly in writing and given information, assistance and the sole authority to defend or settle same at XYZ's expense, and XYZ Ltd shall pay all damages and costs finally awarded therein against Buyer. In case said product is in such suit held to infringe and the use of said product is enjoined or in the case of a settlement as referred to above, XYZ Ltd shall have the option, at its own expense, to procure for Buyer the right to continue using said product; or modify same so it becomes non-infringing; or refund the depreciated value of said product and accept return of same. XYZ Ltd shall have no liability for any infringement of patents of trademarks or copyright resulting from compliance with Buyer's designs.”

*The Australian Computer Journal, Vol. 19, No. 4, November 1987*
Ref: 8 — Disputes
Most vendors will accept an independent arbitrator to help settle disputes. In all cases, insist that the law governing the contract is under the statutes of at least one of the Australian states. In one contract signed at an Australian educational institution the following was agreed to:

“This Agreement shall be construed in accordance with and governed by Scots Law and the Licensee agrees to submit to the jurisdiction of the Scottish Courts.”

This is not satisfactory especially as the software was obtained from an agent in the USA.

4. CONCLUSION
The reader will now be aware of many traps and pitfalls to be avoided when purchasing software. Perhaps you have had experiences of contracts with clauses even less satisfactory for the purchaser than the ones we have encountered. If so, why not write to the editor so we can share our knowledge?

5. ACKNOWLEDGEMENTS
I wish to acknowledge the assistance of Graeme Knox (Computer Centre Manager, RMIT), Shirley Mole (Legal Officer, RMIT), Joy Geary and Gabrielle Eager.

BIOGRAPHICAL NOTE
Norbert Riedl, BSc, MSc, MACS is presently Manager of Administrative Computing at RMIT. Before taking up this post he was a lecturer in the Department of Computing, RMIT.
Non-professional Practices in Computing: Some Thoughts on the Next Decade or So

R.A. Coldwell†

Copyright © 1987, Australian Computer Society Inc.

INTRODUCTION
The Australian Computer Society republished an expanded Code of Ethics in 1985 (ACS, 1985). This was a statement by a Society which was becoming aware of the dangers of letting computer-based crime spread in the way that was threatened. Its intent to maintain a Code of Ethics to promote the integrity of persons engaged in the practice of computing received some wry smiles in computer rooms throughout the country. Yet, during the following year, Benbow et al. (1986) found that Australian Electronic Data Processing Managers had little idea what to expect regarding computer crime. Later still, Costello (1987) expressed the opinion that modern society attaches varying degrees of gravity to different crimes, and that down near the bottom of the list are the white collar crimes. He placed computer crimes, for example, between car theft and that vast array of laws surrounding the use of motor vehicles.

Others have described, elsewhere, various phenomena related to computer crime. The popular press goes to great lengths (discussed by Costello, 1987) to give it adequate coverage and yet, apart from readers’ concerns about how many dollars those crimes involve, their response is often expressed in terms of glee. Yet Burchall (1987), discussing the anonymity of computer crime generally, points out that although computer crime is seen as victimless it is costing the community hundreds of thousands of dollars a year. Others would consider that to be a conservative estimate.

Computer professionals — ordinary human beings that we are — have been heard to discuss these phenomena in impartial terms. Indeed, a respectable member of the Australian Computer Society mentioned, recently, that computer crime should not be the concern of the Society.

Rather, the industry provides the hardware and software; what criminals do with it is their own business. The suggestion that some of the criminals concerned might be members of the Society received a blank stare.

Indeed, computer technology is available to those on both sides of the law and below two instances will be given to demonstrate criminals' ingenuity concerning computer mis-use, and how highly structured computer-criminal society is becoming. It is apparent that there is much fiction, which is being portrayed in the media as fact, written about both computer crimes and computer criminals. Alas, this is often reinforced by intelligent people without their realisation. Fitzgerald (1986) had commented that one group which have attracted criticism are students on work experience. They are often reported as having the motivation and the means to cause problems. To this an eminent Australian academic replied that hacking can be done from anywhere in the world. (The police are subdivided by state and national boundaries whereas criminals of all kinds are wise to ignore them.) Yet his unthinking response implied that "It's not one of us; it's one of them". If it is not computer professionals involved in computer crime, who is it? Implying that Australians don't do things like that is somewhat naive.

COMPUTER CRIMINALS

Computer criminals are often portrayed in the media as spotty youths sitting in darkened rooms working by the light of cathode ray tubes. Their parents are, no doubt, snoring in bed while their sons — they are invariably portrayed as boys — are sitting in the early hours of the morning leering at the screens of their microcomputers.

What proportion of computer criminals, who are caught, are children? It is not known because reliable statistics on computer crime are only available to the police, and not to the computer industry. As reported, the proportion appears to be quite high but, it should be remembered that children's crimes sell newspapers. How representative is the high incidence of child crime reported in the media? Are those, who are learning their craft,
caught more often than their elder brothers or, indeed, their fathers? Bearing in mind that this is a white collar crime, can we consider their mothers and sisters as potential computer criminals? It seems to be so, but there are no answers to these questions and no reliable statistics available on computer crime. Even those given in research papers are incomplete.

In crime, in general, there is a dichotomy formed by individual and group-based crime. Computer criminals are perceived of as working alone. Is this an unjustified assumption? Many criminals are employed by criminal organisations to work together in groups. Can this generalisation be extended to computer crime? Indeed, if the police are to have any success in apprehending computer crime, common types of computer criminal should be identified and crime should be modelled as dynamic situations (see Coldwell, 1984, 1985). There is a dearth of reliable exploratory research in this field and little research funding.

### SOCIAL MODELLING

From close content-analysis of 5,078 articles from newspapers of different countries (see Table 1), some surprising trends in our beliefs about computer-criminals, their organisation and their sources of employment have been formed:

(i) Computer-criminals are portrayed and stereotyped as either children, youths or teenagers (see Benbow et al., 1986).

(ii) Computer-criminals are thought to hack for fun or for intellectual stimulus.

(iii) There is no recognised concept of planned infiltration by organised criminal bodies into other organisations (see McNeil, 1978).

(iv) We have not come to terms with the concept of an existing criminal world becoming computerised (see Petersen, 1986).

(v) We are more concerned about hacking related to the world of commerce and less about defence security.

(vi) We do not recognise that there is an international circuit of expertise in computer crime which can be hired, simply at a price, by any organisation to do a job.

(vii) The focus of computer crime seems to be concentrated deliberately at a parochial level to avoid publicising corporate computer-based crime.

(viii) There is disbelief amongst members of the computing world that people elect to work with computers to avoid working with other people.

There is a strong tendency for computer criminals to be stereotyped by the press. Watson-Munro (1986) describes some real psychometric properties of white-collar offenders as above average IQ, low in self-esteem and poor in communication skills, with a strong desire to succeed and to project an image of success, poor financial management skills, popular with peers, superiors and subordinates and well-regarded within the domestic community. This model does not fit what the media portrays as a computer criminal stereotype. He goes further to say that often, in times of stress, these individuals, despite their better than average intelligence, are unable to cope with their environment and to objectively appraise alternative and legitimate means of dealing with their problems. Robinson (1987), meanwhile, suggests that the higher echelons of hacking have a total disrespect for authority and do have something missing socially. In advising employers, Murchie (1987) writes that likely employees to watch for are those who arrive 30 minutes early, do not take a dinner break, stay late and play with computers at home. Indeed, Macken (1987) emphasises that patterns in computer crime should be examined. Beyond all else, there seem to be social patterns of the sort outlined above which could be developed into dynamic social models of computer crimes.

### ORGANISATION OF COMPUTER CRIME

Computer crime is not disorganised. Computer-criminals are no less organised than non-computer criminals (see McIntosh, 1975). Various writers have described the extent to which criminal society is utilising high technology but two references will suffice here. Petersen (1986) described, for example, how brothels in a number of large US cities have had IBM PC-AT computer systems connected to each other via modem links. He explains that male visitors to New York, who request service, will have their records from bothels in, say, Houston examined before a prostitute is briefed. Concerning an insight into the motivation and organisation of hackers, Robinson (1987) writes that if new hackers get into a bulletin board they have to earn brownie points. They have to give information to get information. He describes a board with a mediaeval hierarchy system of serfs and kings, and claims that they are just playing out their egos. It is also apparent, from the writings of Landreth (1985) and Cornwall (1986) about hacking, that hackers express a high degree of arro-

---

**Table 1. Content Analysis of Newspaper Articles**

<table>
<thead>
<tr>
<th>Major Topic of Article</th>
<th>No.</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warnings to commercial organisations</td>
<td>721</td>
<td>14.2</td>
</tr>
<tr>
<td>Exploits of children and youths</td>
<td>528</td>
<td>10.4</td>
</tr>
<tr>
<td>Technical sophistication of hacking</td>
<td>467</td>
<td>9.2</td>
</tr>
<tr>
<td>Lack of reliable advice</td>
<td>392</td>
<td>7.7</td>
</tr>
<tr>
<td>Police confusion about trends</td>
<td>366</td>
<td>7.2</td>
</tr>
<tr>
<td>Use of passwords as deterrent</td>
<td>261</td>
<td>5.1</td>
</tr>
<tr>
<td>Legal deterrents to computer crime</td>
<td>197</td>
<td>3.9</td>
</tr>
<tr>
<td>Banks as targets for hacking</td>
<td>171</td>
<td>3.4</td>
</tr>
<tr>
<td>Confidentiality as deterrent to reporting</td>
<td>156</td>
<td>3.1</td>
</tr>
<tr>
<td>New technology of computer crime</td>
<td>121</td>
<td>2.4</td>
</tr>
<tr>
<td>Attributes computer crime to disadvantaged</td>
<td>102</td>
<td>2.0</td>
</tr>
<tr>
<td>Consultancies set-up to combat trends</td>
<td>102</td>
<td>2.0</td>
</tr>
<tr>
<td>Activities of national security bodies</td>
<td>101</td>
<td>2.0</td>
</tr>
<tr>
<td>Student-based computing activity</td>
<td>99</td>
<td>1.9</td>
</tr>
<tr>
<td>Arson attacks</td>
<td>99</td>
<td>1.9</td>
</tr>
<tr>
<td>Organised crime's use of hacking techniques</td>
<td>97</td>
<td>1.9</td>
</tr>
<tr>
<td>Space installations as targets</td>
<td>96</td>
<td>1.9</td>
</tr>
<tr>
<td>Threat to national security</td>
<td>96</td>
<td>1.9</td>
</tr>
<tr>
<td>Educating police to combat hacking</td>
<td>94</td>
<td>1.8</td>
</tr>
<tr>
<td>Others</td>
<td>812</td>
<td>16.0</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>5078</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
gance, that their response towards hacking is amoral and that they regard the technological innovation involved as sufficient to justify a crime which they do not acknowledge.

One cannot help but muse, as Cooley (1980) does, on the comment of the Nazi Albert Speer whom he quotes as having written “Basically, I exploited the phenomenon of the technicians’ often blind devotion to his task. Because of what appeared to be the moral neutrality of technology, these people were without scruples about their activities”. Indeed, Stanley (1985) points out that even to the general public computer crime is often rationalised as hurting no one directly.

CORPORATE COMPUTER CRIME

Bearing in mind the apparent peer group pressures that hackers are subject to, how likely is it that groups of them work in teams? Bearing in mind, too, the benefits to be gained from successful hacking, might they not also be organised by criminal bodies? Indeed, as Wassall (1986) points out, a special team of highly trained officers is now in charge of tracking down Britain’s computer crooks, and we can assume that the FBI, the CIA and other official bodies have also formed teams. But, having formed them, how are those teams being used? Macken remarks, for example, that if the tax computer was linked with the bank’s computer, the tax office would discover all the undeclared interest earned on savings. International security bodies appear to have similar access, making the fuss about an Australia card appear insignificant.

A possible scenario suggests that under the cover of corporate bodies, there are teams of hackers working, infiltrating and, indeed, invading one another’s privacy. If the records from that brothel network were made accessible to others, for example, thousands of people might become subject to blackmail. Whereas the man-in-the-street clearly considers financial gain to be the aim of hackers, people who employ hackers may have more devious uses of the information in mind. Power over peoples’ actions, through blackmail and other misuses of knowledge, is more profitable (and certainly more disruptive) than obtaining a few thousand dollars by hacking. Yet this is ignored by the press.

But what evidence exists that corporate crime might be rife in Australia? Recently, Lowe (1987) wrote in The Australian about an unsuccessful approach to the Criminology Research Council for funding for an investigation of computer crime. ACARB received 27 calls in response to the article; seven of these were from insurance companies, four were from banks and four were from foreign embassies all of whom, curiously, asked questions related to whether evidence of corporate computer-based crime by otherwise respectable organisations existed (see Table 2).

TERTIARY EDUCATION

Many of the computer criminals who have been caught have been children. We have reports that some of the teenagers, who have been apprehended, have since been employed by the FBI and other such organisations. The journalist, in each case, implied that they were employed to counter hacking. But how credible is this, bearing in mind the aggressive function of such bodies?

Many students are being educated in Departments of Computing in tertiary institutes throughout Australia and elsewhere. An examination of one such course structure to see what evidence there is of effort to counter any tendency for a spread of a technician mentality worldwide in the computer community reveals that the predominant subjects in the course (RMIT, 1986) are programming, computer architecture, systems analysis and applied scientific and commercial programming techniques, data base technology, compiler writing, software engineering and data communications. These are combined with a selection of any four subjects from a list which includes as subjects personal identity and community, science, technology and social responsibility and power: its use and abuse. This suggests that, in one tertiary institute at least, some attempt is made to establish a man-machine ethic.

What are the characteristics of people who choose to work predominantly with machines instead of working with people? The “extrovert” and “introvert” stereotypes are familiar to us from psychological theory. Most professions have requirements which tend towards one or other extreme. Those who choose the arts, it is said, tend to be extrovert whereas those in the sciences tend to be introvert. True, these are vague categories but, in dealing with a steady flow of students, this bias is noticeable as far more than merely a trend. Those who are good at computing tend — as Watson-Munro suggests of computer criminals — to be of above average IQ, low in self-esteem and poor in communication skills with a strong desire to succeed and to project an image of success, poor financial management skills, popular with peers, superiors and subordinates and well regarded within the domestic community. Most computer professionals, the writer included, possess some, at least, of these negative attributes.

THE NEXT DECADE

In conclusion, the scenario can be developed into something more dynamic, and used to attempt to predict what might occur within the next decade worldwide.

When McIntosh (1975) wrote about the organisation of crime, a little over a decade ago, she made no mention of

Table 2. Topics Mentioned in Series of Enquiries re Computer Crime.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Frequency</th>
<th>of mention</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incidence of attack</td>
<td>27</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Techniques available to foil attack</td>
<td>23</td>
<td>21.3</td>
<td></td>
</tr>
<tr>
<td>Corporate-based computer crime</td>
<td>15</td>
<td>13.9</td>
<td></td>
</tr>
<tr>
<td>Screening of employees</td>
<td>14</td>
<td>13.0</td>
<td></td>
</tr>
<tr>
<td>Attack by government agencies</td>
<td>11</td>
<td>10.2</td>
<td></td>
</tr>
<tr>
<td>Research into computer-based crime</td>
<td>7</td>
<td>6.5</td>
<td></td>
</tr>
<tr>
<td>Others</td>
<td>11</td>
<td>10.2</td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>108</strong></td>
<td><strong>100</strong></td>
<td></td>
</tr>
</tbody>
</table>
computer-based crime. Her only reference to the use of technology was a comment that the more advanced the techniques and the larger the technical skills, the more difficult it will be to prevent racketeering. Little did she and other criminologists realise what would happen during this next decade. Indeed, criminology finds itself confused by the use of high technology by criminals. Criminology, almost since Conan Doyle’s time, has veered towards the social sciences and is, at present, embarrassed by its lack of knowledge of computing. It could be argued that Doyle’s Sherlock Holmes had a mind closer to that of McNeil’s (1978) hero in “The Consultant” than has ever been conceived by a modern-day criminologist.

The current phenomena stem, historically, from the innovative hacking of various Italians in the early 1970s and its development by Italian Americans during the following decade. The writer was advised to learn two languages concerning the monitoring of computer crime, Cobol and Italian. Since these early developments, the skills have spread worldwide and are alive, well and in use in Australia. However, computer-criminals are conceived of as high-tech, blundering amateurs. Our attention is focussed, almost daily by the press, on relatively low-key, white collar offenders. Yet the criminal sector is, like the rest of society, becoming computerised rapidly. Further, it has a profession of people, with machine-related beliefs, to call on for expertise. Some of these people are, inevitably, arrogant, amoral people who crave self-esteem and are, therefore, relatively easy for the criminal world to manipulate. The police, by comparison, are massively disadvantaged in Australia, where criminal activity is treated something like a sport. It would be unfair to give one side an advantage over the other and it is considered bad form to rob your own superannuation fund.

A massive leap forwards and upwards in computer crime can be anticipated over the next few years. It would be naive to assume that the ultimate aim of hacking, for example, is financial gain; it will be raw power. As high-jacking jet planes was a target of terrorists, during the last decade, playing computer chess with the world economy and strategic defence systems may be the next goal.

Whereas it would be irresponsible to publish a full scenario, concerning both computer-vandalised defence procedures and computer-created economic chaos, it would suffice to say that these two areas, vandalised in combination, could easily return the world back to the middle ages.

REFERENCES


BIographies NOTE

Dr Roger Coldwell is a Sociologist and a Research Associate of the Australian Computer Abuse Research Bureau, Royal Melbourne Institute of Technology.
A Guide to Software Quality for the Developer and Unwary User

Roger Farrell†

Quality assurance methodologies have been part and parcel of the manufacturing environment for many years, and over the past few years have even greater emphasis. It is now becoming obvious that in order to make better use of the limited resources we must apply these disciplines to the software development process. Stowe’s R&D experience over the past four years has shown that it is possible to provide users with software with a minimal number of defects and that the cost of doing so is much less than if the techniques are ignored. Keywords and Phrases: quality assurance, quality measurement. CR Categories: C.5, D.2, D.2.1, D.2.2, K.6.1, K.6.3.

INTRODUCTION

The importance of software quality is being increasingly recognised by both the users and the developers of software. Whenever software is purchased, the quality is a major factor in the selection. An impression of this quality of software is formed from examining the software and from its reputation. Quality software is expected to:

— meet the user’s requirements
— be reliable
— be relatively free or errors
— be usable
— be value for money

The cost of poor quality software is well known. It is the exact opposite of the benefits that can be obtained from software that satisfies the factors set out above. Most users are familiar with acquiring software that does not meet their needs or that contains numerous defects. Similarly, most developers are familiar with fixing defects in software many months, and sometimes years, after the software has been acquired.

Given a choice, users naturally choose high quality software in preference to software that has a reputation of poor quality. The problem is that this choice is often not exercised, as the quality of the software appears difficult to assess. In addition, there appears to be no widely used method for controlling quality while developing software. The problem is further compounded by the view that software is often a one-time job and therefore quality is an unnecessary frill. However, software development is no different from developing other more tangible products, such as a car.

Similarly, software requires the same criteria of usefulness, reliability and availability that are used to judge other products. Software that does not meet these requirements, contains numerous defects and is inoperable while the defects are being fixed, will have a very short life span. While good management and adequate methods are prerequisites for improving quality, they are not the complete answer. Many well managed software projects, using the best tools and methodologies, have failed to produce quality software.

The criteria for the successful development of most software products to date has been whether or not the task was completed on time and within budget. Typically, quality has been ignored, or at best checked at the end of the development cycle, with no attempt to plan or measure the level of quality. Developing the lowest cost software, irrespective of quality, is simply becoming unacceptable to the user.

There is no reason why the mechanisms used to achieve quality on more tangible products cannot be applied to the development of software. This concept is certainly not new, as IBM, Boeing, NASA and some government bodies have been using this approach for over 10 years.

However, despite this and a considerable amount of published material on software quality, very few organisations today have any workable practices for monitoring and controlling software quality. Two issues derive from this. The first is that there is a need by software developers to recognise that they are manufacturers and introduce the disciplines of the manufacturing environment. The second is for users to recognise the problems they face in accepting software that has not had a quality assurance methodology applied to it over the full development cycle.

QUALITY ASSURANCE BENEFITS

The introduction of a quality assurance methodology will address the situation by providing workable procedures and practices that can be used to increase the quality of software. The central theme of this methodology is that software quality can be measured and that this measure can be used to:

— establish software quality goals
monitor the quality of software both during and after development
— take actions to alter the level of quality
Theoretically, the objective is to produce zero defect software. In reality, a more practical objective is required. There are at least two reasons why a less ambitious goal is necessary:
— The quality goals must be attainable. If the goals are too high, failure is almost certain and the exercise will cease to be a serious attempt to control quality.
— The quality goals must be guided by economics. The cost of poor quality software over its life is usually high due to the effort required to maintain the defects. By applying the methodology to the development of software, the overall costs can be reduced due to less maintenance. However, there comes a point at which the costs of applying further quality measures outweigh the benefits. That is, the project or product must also have a positive return on investment within an acceptable time frame.

The measure of software quality as we have defined it, is simply the ratio of the number of defects per thousand lines of code. The number of defects is usually difficult for most people to formalise, however it is a well defined concept and, once understood, is not difficult to identify. There should be no disagreement about what should or should not be classified as a defect. The following definition may help clarify this:

“A defect is any flaw, error, ambiguity or incompleteness found while developing, testing and using the software.”

This includes defects detected after the software is in operation as well as defects detected during development in the specifications, in the test data or results, in user manuals and even errors found in the program comments and help messages.

In short, a defect is any deviation from perfect software. That is, it is any deviation from developing software that is 100 per cent correct the first time. In most software developments, the number of defects, as defined above, is simply not known.

By comparison, the other factor in the quality measurement ratio, lines of code, is quite simple. Lines of code includes all executable statements as well as data definitions and any other statements that affect the creation or operation of an executable module. It excludes comments and compiler options for formatting compilation listings. Lines of code is often abbreviated to LOC, with KLOC representing a thousand lines of code.

Past experience in Stowe’s Research and Development Division has shown that software typically contains 60 defects per KLOC. Of these 60 defects per KLOC, software developers can expect to remove 50 during development without quality assurance procedures. The remaining 10 defects per KLOC are usually detected during the operational life of the software. For a product of 100 KLOC, this represents 1000 undetected defects in the software and documentation at the time the user accepts and implements the software.

Using the methodology, however, 57 of the 60 defects per KLOC can be expected to be removed during development, leaving 3 defects per KLOC undetected. This results in only 300 defects remaining at implementation time for the same 100 KLOC product.

As more expertise is gained a figure of less than 1 defect per KLOC remaining on implementation can be expected. Using this figure, less than 100 defects would remain in the 100 KLOC project.

The quality goal then, is an estimate of the number of defects that will be detected from the start of the development through to the end of the software’s operational life. To be able to estimate this figure, a number of questions need to be answered:
— What is the current level of quality in the software developed, without using a quality assurance methodology?
— When will the defects be detected in software developed?
— How many undetected defects will remain in the software developed, when it is implemented?

To answer the second question, the following figures provide an estimate of the number of defects detected, and when they are detected:

6 defects during preliminary design inspection 10%
15 defects during detailed design inspection 25%
24 defects during coding inspections 40%
6 defects during unit and function testing 10%
6 defects during system testing 10%

These figures represent 57, or 95 per cent, of the total 60 defects per KLOC being detected during development. This leaves 3 defects per KLOC to be detected during the operational life of the software.

This is the quality measure that matters. This is the quality that the users see. For this reason, the quality goal is usually expressed in terms of the remaining defects per KLOC at the time the software is implemented. Some of the best available software have defect rates in the range 0.1 to 1.0 defects per KLOC on implementation. This level of quality can, and has, been achieved. Levels in this range, and even up to 3 defects per KLOC, are not commonly achieved without a quality assurance methodology. Typically, defect rates in the order of 10 defects per KLOC are recorded for software developed without such a methodology.

MANAGEMENT ISSUES
Turning to the management issues surrounding quality assurance, we must look at the two prime factors involved — costs and benefits. Over the total life span of software, the cost of developing quality software is far cheaper than developing software without quality assurance. The earlier defects are found, the easier and cheaper it is to fix them.
By addressing defects early, the impact of defects rippling through the system is minimised. During the development period, the quality assurance procedures both add to, and subtract from, the development costs, with no overall increase. However, when the software is operational, lower maintenance costs start returning dividends when compared with conventionally developed software.

Quality assurance adds approximately 10 per cent to the development costs incurred during the functional specification through coding stages, as a result of implementing the key plans, including the quality plan, reviews and inspections, and the setting up of a test methodology. However, quality assurance then reduces the costs during the unit, systems and acceptance testing stages by the same amount. In addition, it ensures the maintenance costs, after implementation, are reduced in the order of 40-60 per cent. The net result is an increase in quality and a reduction in total costs.

The most significant benefits are the financial ones of reduced costs and increased quality. Other benefits that result, include:
- Guaranteed satisfaction of user requirements
- Maximum software reliability and integrity
- Increased staff productivity
- Decreased maintenance effort
- Increased statistical data for future projects

In addition, there are many direct benefits for the development staff, including:
- Improved staff skills through education and training
- Improved motivation and morale

which all lead to improved development staff satisfaction and a more productive team environment.

CONCLUSION
Users of software need to recognise that the product they are receiving from the developers is flawed and apply pressure for the correct methodologies to be used in its manufacture. In the interim, until the benefits begin to flow through and the defect levels are significantly reduced, internal users should apply more stringent acceptance testing of both the software's usability and the documentation provided to support it. This is a little like closing the stable door after the horse has bolted, but it will go a long way towards preventing the defects from causing major difficulties in the day to day running of the organisation.

In that other area of usage, the packaged solution, prospective purchasers can estimate the quality of the software in the following way. Firstly they should ask for an extensive list of current users from whom a view of the history of the package can be obtained. Secondly, they should insist on trying before buying, and following the advice given above in setting up and conducting a comprehensive and exhaustive acceptance test. Another indicator of the quality of packaged solution that is available to the prospective buyer, is the documentation provided with the product. The number of defects found herein both the written word and in comparison with the operation of the product, will provide a good indication of the overall level of quality.

The number of defects in software can be significantly reduced. It requires both pressure from the users and a commitment by developers. Without these two activities, we will continue to wait longer and longer, and pay more and more, for our software solutions as the burden of corrective maintenance swamps us.

FOOTNOTE: A description of Stowe's software quality assurance methodology was presented at a recent conference (Farrell, 1987). This methodology permits the development of quality software and the avoidance of defects.

REFERENCE

BIOGRAPHICAL NOTE
Mr Farrell is responsible for the development of the TCS package at Stowe, and has over the past few months, been implementing the quality assurance methodology developed by Stowe's Research and Development Division. This methodology has resulted from four years of development activity with IBM at its software laboratories.
In this article, Roger Clarke offers us a short list of those books which every information systems practitioner and teacher should have on their bookshelves. He has been assisted in his compilation by many colleagues from Departments around Australia teaching Information Systems.

Keywords and Phrases: computing professional, essential booklist.

CR Categories: K.7.m, H.o.

SOFTWARE DEVELOPMENT

Analysis — Procedure-Oriented


Analysis — Data-Oriented


Prototyping Approaches

Analysis — Comparative Methodologies


Design


Construction


Implementation


Maintenance


TECHNICAL TOPICS

Quality Assurance


Audit and Security


Systems Software


Management Information Systems


Decision Support Systems

Data Dictionary

Database Systems

Free-Text Database Systems

Communications and Networking

Office Automation

Software Development Environments

Knowledge-Based Systems

Natural-Language Processing/Human Thought

MANAGEMENT
Operations

Project Management

Software Management

Psychology

User Participation

Information Systems Management

Legal Aspects

Social Implications

Philosophy

REFERENCE

MAJOR PERIODICALS
Australian Computer Journal.
Communications of the ACM (US).
Computer Journal (UK).
Computing Reviews (US).
IEEE Computer (US).
Information and Management (US).
MIS Quarterly (US).

TEACHING AND TEXTS

BIOGRAPHICAL NOTE
Roger Clarke is a Reader in Information Systems at the Australian National University. He had wide professional experience before taking his current academic post.

The Australian Computer Journal, Vol. 19, No. 4, November 1987
The need for an adequate continuing-education programme is considered from the points of view of professional standards, career motivation and the requirements of the computing industry. Consideration is also given to methods of course presentation. Four areas of high priority are identified.

Keywords and Phrases: professional development, postgraduate education, software export.

1. INTRODUCTION

Within the Australian Computer Society, a cornerstone of policy requires members to obtain and then maintain a broad knowledge of computing technology, in addition to skills in their area of speciality. Professor Northcote (1987), governor of the membership board, states “To maintain any semblance of competence, it is essential that a computing professional undergo extensive continuing education”.

In addition, the Information Technology industry must promote continuing education in order to sharpen its competitive edge.

Between October 1986 and March 1987, a study of educational needs, as perceived by computing practitioners, was carried out in Western Australia (Woodings, 1987). It included professional and career path needs plus the knowledge prerequisites for a viable computing export industry. Attention was also paid to methods of course delivery.

2. QUALITY OF COMPUTING INDUSTRY PERSONNEL

The computing industry is healthy and boasts a number of successful companies producing mainly software. Australian tertiary institutions are producing good quality graduates who are employed within a few weeks of receiving their qualifications. In addition, there is a steady stream of migrants with computing qualifications from Zimbabwe, South Africa, Britain and South-East Asia available, so that none of the medium to large companies report any great difficulties in filling vacancies. However, many recruits do not have the full range of required skills or possess all the necessary experience. Some managers report a higher turnover in staff than is desirable. Fortunately, people find Australia an attractive place to live and it has fewer staffing problems than other western nations.

There is no apparent shortage of people with innovative ideas and WA has something of an entrepreneurial reputation. But there is little evidence that computing companies there do any better than the national average, with only 5% of gross income coming from exports (DITAC, 1986). Most companies appear to be finding sufficient work in servicing the local industry. Recently there has been great interest in the production of defence systems. A report on defence prospects is being produced by the WA Department of Industrial Development (1987). Unfortunately, few in WA have experience of very large projects (over a million lines of code) or of ADA, which is a large language requiring experience for efficient use. A possible problem area is leadership. Have managers been promoted from the programming ranks too early without gaining sufficient management skills? Are they too conservative; unwilling to embrace technological trends which have emerged since they received their undergraduate training? Whatever, the full potential of this important high technology industry is apparently not being realised.

3. SURVEY METHODOLOGY

The objective of the study was to determine and evaluate ways of improving the quality of computing practitioners. It was concerned with two overlapping issues: how could the skills computer programmers obtained as undergraduates and during early work training be most effectively upgraded in the face of rapidly changing technology, and how could their ability to produce innovative systems be enhanced in order to improve the competitive position of the industry. These two issues are highly subjective. Because of this and because of tight time and budget constraints, it was decided to collect information by interview rather than questionnaire.

With the need for Australia to depend less in future on primary industry and more on the so-called “sunrise technology” now clearly identified, there have been a number of
reports commissioned in recent years to investigate abilities and markets. Of particular note are:

- National Technology Strategy — A Discussion Draft, produced by the Department of Science and Technology (1984) for the National Technology Conference. It suggests targets for the budgeting of Research and Development as a proportion of Australia's GDP but only mentions in passing the education of professionals. It suggests that efforts should be made to increase the proportion of school leavers entering tertiary education to 20% by 1995. As the corresponding proportion in Japan is currently about 70%, this is rather disappointing.

- Professor John Bennett (1986) pointed out a further disappointing comparison. "This country's future will rest on the shoulders of its scientists and technologists. Per capita, Australia has nine times as many lawyers as Japan. But Japan has nine times as many engineers and scientists as Australia."

- Information Technology in Australia — Capabilities and Opportunities (in three volumes), produced by W.D. Scott and Company in association with Arthur D. Little Inc. (1984). It presents a detailed examination of the Australian industry and puts forward five action programs. It comments on the need to increase funding of tertiary electronic engineering and computer science departments but particularly stresses the need to achieve an increase in the number of effective IT entrepreneurs. Skills needed are: business planning, financial control, cashflow management, marketing, project management, production, cost reduction and quality control, all in a high technology environment.

- Report on the Capability of the Australian Computer Hardware Manufacturing Industry — Strategies for Progress (University of WA — CABR, 1986), which gives a comprehensive overview of the local hardware scene. A major area of software production for export is in applications associated with specialist equipment. The Department has also produced The Australian Defence Software Market — A Working Paper (1987) to assist companies wishing to enter this specialist area. Considerable staff training will be needed by such companies.

- The ACS Membership Board has produced a revised syllabus for its examination on the minimal level of knowledge needed as a computing professional (Northcote, 1986). It was completely revised for new examinations in 1987.

4. CURRENT STANDARDS OF KNOWLEDGE

The ACS has defined a basic level of knowledge, which it considers the minimum necessary for professionals in the computing field, in the syllabus for its entrance examination. It has four sections: Basic Computer Concepts, Programming Techniques, Systems Analysis and Design, Data Management and Communications. The syllabus represents about one year of full-time study for an average student. The Society considers that anyone who does not have a good grasp of the fundamentals of the above four areas should not be considered a computing professional. That is, there would exist doubts as to the validity of advice they might give on computing matters or on their ability to make a good decision on behalf of their organisation.

At the 1981 Census, 30% of computer programmers and analysts had a computer science qualification (Krieger, 1986). Since 1 January 1986, the ACS has insisted on a tertiary qualification or a pass in its examination from all candidates seeking membership. However, as is likely in a field with such a short history, there is a large group of members who gained their knowledge the hard way — through doing, and learning from their mistakes. Their experience represents a valuable pool of knowledge. Some later take formal courses, reasoning that there is no substitute for a comprehensive and well balanced course of study.

Although definitions of computing professional may vary, it is evident that the ACS only represents about half the professional people working in the IT industry. That is not to say that the non-members are unqualified, inexperienced or act in an unprofessional manner. It does mean that they are attempting to maintain their skill or widen their knowledge through other than ACS avenues (journals, meetings, annual conferences, Professional Development seminars or Special Interest Groups).
Some companies have written into their staffs' duty statements an allocation of time (often 5%) for reading journals or text books. Some also prepare competency profiles on staff seeking promotion. In addition to basic skills, these profiles include specialist training which might be needed for the higher position. A few companies produce annual training schedules for their staff. These are useful policies for encouraging the continuing education of staff, but are not widely used.

In the Nash (1987) study, by far the most frequent comment on the quality of new trainee programmers concerned their lack of communication skills. One possible reason is that many students chose to enter computing and engineering, rather than medicine, law or teaching, because they enjoy problem solving and technical complexity rather than interacting with people. They are not necessarily good at verbal or written communication skills. These skills are not particularly easy to teach in a theoretical fashion and most tertiary computer science departments do not give a high priority to these areas. They are skills often best acquired in the work environment. Similarly, management skills are not taught to new programmers as they are unlikely to need them for several years. Unfortunately when promotion to a managerial level is sought, the applicant seldom returns to University for a comprehensive course.

The trend over the last two decades is for employers to look more to recruiting computing staff from outside rather than the in-house promotion of clerical or technical staff. Applicants with a degree or diploma are preferred (although experience only is still acceptable, particularly in small companies). All organisations pay at least nominal attention to ongoing education.

At some point in the next decade, it appears likely that some form of professional registration of computing people will be implemented. Moves in this direction have already been made overseas, notably in Britain when it was recognised that data bases, improperly administered, could inflict profound damage on society. Two avenues are possible: a government implemented register, covered by an act of parliament, such as we have for doctors, nurses and lawyers; or one organised by the ACS in a similar manner to the psychologists and accountants. The latter scheme requires substantial advertising to make employers and the public aware of the advantages of employing a registered member. Either way, a scheme must then be implemented to ensure that members on the register make a sufficient effort to maintain their knowledge level.

During the study, one feature became plain, that attendance at courses is usually initiated by the staff member rather than his/her manager. Further, it is probably done with an eye to the course attendant's future career rather than the specific good of the company. The main motivation comes from peer-pressure. If fellow graduates are getting ahead of you it is probably because they know more than you do.

In summary, the proportion of professionals in the industry with a tertiary qualification in computing is slowly increasing. After entrance, many only pay lip-service to the principle of continuing education — most training being product specific in response to short-term needs. The study of computing principles underlying some new technology is often initiated as part of a career path. Many professionals do not have a high level of personal-development skills such as planning, organising, or documentation. Many do not appear to attach much importance to such skills.

5. CURRENT PROBLEM AREAS

A number of problems in the computing industry were discussed in section two. Similarly, some problems relating to the education of individual members of the industry were given in the previous section.

Two points are worth stressing: we must improve the general skills of all practitioners if the industry is to gain a reputation for the production of superior systems, and second: we must identify those brighter individuals with good ideas so that they may be supported by teams who do have all the necessary skills to bring the ideas to fruition. “A new high technology device usually becomes available in the marketplace because of a number of competent engineers and technologists have been doing their jobs properly, and not because of the brilliance of the lone inventor” (AAPCS, 1986).

Overseas, computing companies have set up private research organisations to ensure that they can attract and keep the brightest new graduates by supplying expensive research facilities. In Australia, because of its smaller industrial base, the concept of research fellowships seems more appropriate. Good honours graduates should be encouraged to work for several years in industry before returning to a University to do higher research in a specialist area. Both the work experience and postgraduate research should be under a single firm contract. The advantages are fairly obvious: the individual gets a salary and the freedom to pursue his ideas in a strong research environment; the company obtains a favourable recruiting position and the pick of the bright ideas; and the university benefits from the technology transfer. In 1984, Computer Science departments in Australia gained an average of less than $20,000 in support from the private sector, of which only 17% came from Australian companies (AAPCS, 1986). Such private support is being encouraged by Government through offsets and taxation policies. Postgraduate research fellowships and facilities are an effective way of increasing the level of support.

In both the private and public sectors, there are good teams producing systems for a single local need. At the start of a system's life cycle when a need is established, the first step is to survey the market to see if there are existing products capable of filling the need. This same survey should reveal any existing market place (and a price structure) for the local system, should it be produced. At that point, a decision must be made as to whether to produce a system designed purely to meet the short term local need or to go for a generalised system capable of being marketed. Experience suggests that fully generalised systems with proper documentation cost five times more to pro-
duce. However the advantages are obvious: a raising of the reputation of the company/department involved, a far superior product for the local users and given that the right decisions have been made, a good return on investment. Such an approach, using one project out of the dozen or so which may be under development, represents a good way for companies who have been purely user-demand driven, to improve their product quality and start widening their marketplace. The prime need is a workshop for decision-makers on guidelines for choosing projects worthy of general development.

6. COMMENT ON A CONTINUING EDUCATION SYLLABUS
In 1968, the Association for Computer Machinery (ACM) in the USA produced a Computer Science curriculum. This is constantly under review (McCracken, 1987) and has been revised a number of times. It has become one of the foundation stones for tertiary computer science courses throughout the world. More recently, attempts have been made to devise curricula with more emphasis on the social and business aspects.

Computing practitioners require continuing education courses in six areas:
— Standard computing theory which has changed as a result of emerging technology. Five years after graduation, a practitioner may need considerable new teaching about new methodologies, products, languages.
— Specialised disciplines such as graphics, ergonomics.
— Product specific areas such as the operating system for a new machine.
— Areas of applications. For example, medicine and mineral exploration represent highly specialised areas of computing.
— Software engineering has emerged over the last five years as a fundamental part of system production. It covers such areas as software metrics, project budgeting and quality assurance.
— Career skills which will generally include communication and traditional management skills such as report writing.

Fifth generation systems (Luckyj, 1987), software engineering and product management are frequently identified as the topics where most gains can be made. They are the areas most in demand by computing professionals who hope to improve their organisation’s performance.

A proposed continuing education syllabus includes the following nine broad areas:

**Hardware**
General computing practitioners need knowledge to a level where they understand the advantages and disadvantages inherent in the devices. They should also have a general knowledge of operations room considerations such as physical security.

**Communications**
As with the first section on Hardware, it is important for the general practitioner to have a grasp of the fundamentals of computer communications.

**Business Software**
This includes current office automation and database knowledge. Automatic identification systems should also be understood.

**Scientific Software**
Business computing dominates the industry but there remains a need for scientific programmers with a knowledge of statistics, simulation and operations research.

**Other Software Needs**
After graduation, the trainee programmer is often given the task of software maintenance to allow him/her to gain familiarity with company methodology, standards and with large programs. During the next few years, they need to have their practical experience reinforced with the theory of advanced programming techniques. The man-machine interface and graphics techniques should be included.

**Fifth Generation Systems**
Fifth generation research is currently at an early stage, but important fundamental rules have been discovered and may be used in everyday computer systems, particularly in improving the user interface. For example, programming a system to explain what it is doing for the user, will become of paramount importance.

**Social Aspects of Information Technology**
The social effects of IT on society is an important group of topics which are ignored by many computing practitioners. In fact, a case can be made for it being taught in high school and undergraduate courses as a basic necessity for future living. It is included here in a postgraduate professional development syllabus as a reminder that practitioners need to be aware of the issues.

**Software Engineering**
Software Engineering is a broad cover of the research efforts to provide a more rigorous management of computer systems. Its main topics are Scheduling, Resource Management and Quality Assurance.

**Management and Product Development**
This topic emphasises the special computing aspects of traditional management skills. It complements the Software systems.

7. METHODS OF COURSE DELIVERY
There are many course presentation formats available. The choice will usually be made according to trainer availability and breadth and depth of material to be covered. The purpose of this section is to list a number of alternatives together with some comments on their suitability. Three points must be stressed: first, it is necessary to raise the knowledge level of the industry as a whole and therefore wide dissemination of some courses will be needed. Second, the innovatory reputation of the industry

*The Australian Computer Journal, Vol. 19, No. 4, November 1987*
will depend on the efforts of the few with the bright ideas; therefore classes designed to thoroughly explore the depths of a particular topic over a period of time must be designed. Here, technology transfer through offset programs is appropriate. Third, as McCracken (1987) states, teaching computing skills without hands-on practical experience is like teaching people to play the piano without an instrument — computing “has to be learned through the fingertips”.

Although comments will be made on alternatives, the standard format for the immediate future is likely to remain the three-day seminar. This has evolved as the most efficient way of familiarising a group of ten to twenty people with some advanced topic. Shorter courses fail to achieve sufficient depth, longer ones result in a falloff of attendance as people are called back to work to cope with some “crisis”. To spread the knowledge of particular techniques more widely throughout the industry, two ideas are worth considering. The seminars may be videotaped, making the tapes available for loan/lease after some revision and editing. Further a “training the trainers” policy should be encouraged, whereby such seminars are presented to people with good teaching skills who are then given kits to assist them in spreading the message.

A good alternative format, suitable for many topics, is the short (one or two day) workshop. Instead of being passive listeners, participants must become involved in the exploration of a topic. Simulations of a real-world system are useful, as are case studies on the development of particular projects. Although somewhat difficult to manage properly with groups of strangers, the “Structured-Walkthrough” can be effective in workshops.

For technology transfer a series of master-classes is advocated. Here, an expert with substantial state-of-the-art experience in a well defined area would be invited to spend not less than two months giving a series of tutorials to a fairly small invited group of local specialists who already have a substantial knowledge of the subject. Weekly sessions would be appropriate. In between, the ‘student’ participants would be working on projects, trying out the material learned. The instructor would be available for consultations to business and presenting general talks to tertiary institutions, societies or the public. Good private research facilities should also be made available to the visitor. Again, the videotaping of sessions would be useful.

To maintain and upgrade the knowledge of technical people, the lecture series already available at local universities (Northcote, 1986) are suitable. It is suggested that Computer Science departments be approached to represent their daytime undergraduate courses as a series of modules in the late afternoon or evening for people from industry. Examinations could be given although they should be encouraged as the idea is to encourage attendance by members of the industry. Participants or their companies should pay a low rate for the tuition. Costs should not be high as the lectures are a rerun of previously presented material. However there would be extra staff costs and subsidisation by government or from offsets would be necessary.

The above formats all involve formal live presentations of material. Equally important is the preparation of courses for self study. Here, new technology is useful. “Distance education”, such as is provided by the Open University in the UK, the NIES on TV in Australia and by a number of tertiary institutions, has come a long way since the old concept of correspondence courses. It is expensive to produce, but with carefully selected subjects prepared by professionals experienced in the techniques, can be remarkably effective.

Computer-based training is now being widely used and the interactive video-disc has become a useful supplement. Again it is essential that the material be properly prepared and this is very expensive. But for wide dissemination of standard computing topics, this method is most appropriate.

Also worth consideration is the adaption of material for simulating systems on home terminals or PCs. This is particularly suitable for practice in technical skills or in learning about specific products. Many companies and colleges make microcomputers available for students/employees to take home. Many new software products have self-study online tutorials produced as part of the documentation or marketing kits.

The Australian Information Industry Association (AIIA), at its 1986 annual meeting, put forward its own education and training strategy. It considers that training will increasingly take place at the workplace on an as-needed basis. The concept of “programmers workbench” type software having built-in tutorials is very attractive. However care should be taken that programmers do not gather many titbits of information and then consider that as being equivalent to a sound broad knowledge of the subject.

Since 1976, the ACM has produced a series of Self-Assessment Procedures (for example Gourd, 1982) and some are now being republished by the ACS (Northcote, 1987). These are in the form of a test with solutions, notes and references for further reading on a particular subject. They have the considerable advantage of allowing practitioners to privately test themselves, learn that they are not as aware as they hoped, discover what they do not know and where to find out about it. Their aim is laudable, but the proportion of Australians using them is apparently very low. The distribution of these tests is a worthwhile, simple and cost-effective way of raising knowledge levels. Although they cannot indicate that the practitioner understands the solution or where to apply it, such test booklets raise the awareness of the potential obsolescence of an old degree.

Last, as mentioned in section five, there is the need to return practitioners to universities for postgraduate research if we are to encourage the brighter people in the industry to develop new ideas. The provision of industry fellowships is worth consideration.

8. CAREER EDUCATION BOOKLET
An inexpensive method of giving Professional Development Seminars and Workshops a higher profile would be
to combine details of such courses with the career booklets which are distributed by the ACS and other bodies from-time to time. These booklets generally show, frequently by means of a flow chart (Nash, 1987), typical career structures for entrants to the industry from operator or trainee programmer through to consultant or manager. Besides summaries of course content, it is proposed to combine within such a flow chart, details of when a course should be taken to review skills and prepare for the next career step. The third part of the booklet would then supply a calendar of all such known courses being supplied by IT companies, suppliers, tertiary institutions or the ACS. Reviewed annually at the start of the year, it would encourage both employee and management to produce a training plan. Course providers would obtain free distribution of their planned offerings for the year.

9. DETERMINATION OF PRIORITIES
This section is to provide some guidelines on the assigning of priorities to the various possible courses of action devised in the preceding sections. It is based on three axioms: that education programmes are needed both for the short (less than five years) and long-term (twenty to fifty years); that the support of the innovative elite is as important as the raising of the knowledge standards of all computing practitioners; and that community acceptance and support, particularly from industry executives, can be obtained.

In the past, most decisions on priorities were made on the basis of short-term economic or political gain. It is essential that educational programmes should be seen as being for long-term economic security. From the above we can derive four main courses of action:
— Community and industry commitment to educational efforts.
— Immediate support for innovators.
— Long-term improvement in overall knowledge, standards.
— Long-term provision of an innovative environment.

It is probably impractical to attempt any significant improvement in overall knowledge standards in the short-term. The lesson from the spectacularly successful emerging economies of Japan, South Korea and Singapore appears to be the need for deep rooted community acceptance of long-term goals of economic superiority. No ranking of courses of action will be given. Instead, it is advocated that specific programmes that can be effectively implemented in all four areas be given high priority.
— Promotion within the industry (and later the community), of the need for higher professional standards. This may be done by fostering peer pressure or economic pressure. The former is likely to come about by widespread advertising of appropriate courses. The careers booklet as proposed in section eight would be one possibility. To apply economic pressure, some form of professional registration would be useful. As previously discussed, this would have to be government run or at least have strong support from the public sector. Either way, widespread advertising would also be needed, to convince the public and employers that the two levels (professionals and para-professionals) are distinguished by educational and conduct standards. To be registered, one must display a deep knowledge of the discipline and show that efforts are being made to maintain such expertise.
— Short-term programmes for the support of the innovative few. First step is the devising of guidelines on how to recognise the innovator. As previously described, fellowships would be appropriate for encouraging such people. A number of good courses in project management, product development, quality assurance and software engineering are needed to produce suitable teams to support the innovators.
— Long-term programmes for the overall improvement of knowledge standards. Here the stress should be on technical skills and their underlying theory, particularly emphasising flexibility of thinking. The aim is to produce executives of twenty years time with a high theoretical knowledge and the ability to make clear decisions on complex technological issues. Priorities, here, should be on pinpointing a number of areas such as graphics, the man-machine interface or knowledge-based systems and teaching them as a series of master classes. The presentation of existing university advanced computing courses to industry would also be appropriate.
— Long-term programmes to produce an innovation rich environment. The aim is to significantly increase the ability of current programmer/analysts and junior management to produce high quality systems over the next twenty years, influencing their staff to aim for excellence. Here the emphasis needs to be on software engineering and good management practises. A series of seminars, perhaps using case-studies would be a suitable method of presentation.

10. CONCLUSION
The Software Industry appears mainly to be demand driven, that is to be responding only to immediate user requirements. Its full potential is not being realised.

The proportion of computer science graduates in professional positions in the industry is slightly over 30% and increasing. All organisations in the industry pay at least nominal attention to continuing education. Peer pressure and career path plans are given as the main motivating factors for course attendance. There is a distinct possibility of the implementation of professional registration within the next decade and this should stimulate some expansion in the IT education industry.

Besides improving the general skills of all computing personnel in order to raise the industry's reputation, there is a need to identify the small number of people with the good ideas and put them in a strong support environment from the start. Company sponsored research fellowships are advocated as a partial solution to this need.

Six types of continuing education for computing practitioners are identified. Software Engineering and Management Skills were identified as being the most needed.
The suitability of a variety of course delivery formats is examined. Preference is given to the three-day seminar or workshops (especially if video-taped), for most technical and management areas but with a series of two-month long master classes for new technology transfer. Rerunning current university courses is suggested for the upgrading of old computing skills. The need for "hands-on" experience is emphasised for most training. Distance education and CBT are useful in particular circumstances.

To encourage practitioners to plan their professional education to fit their company's needs as well as their career aspirations, a new form of Careers booklet is described.

To assist the small software house to get out into the world market with its products, a workshop designed to show it how to identify which of its current internal projects could be converted into marketable software is suggested.

The final section offers some comments on how priorities might be determined for various options. They are based on the need for community acceptance of higher standards, on the balance between short and long-term objectives and on the need to improve the computing skills of the whole industry versus those of the innovative few.

11. REFERENCES

BIOGRAPHICAL NOTE
Terry Woodings first entered the computer industry in 1963 later gaining a BSc in Statistics and a Diploma in Computing from the University of Western Australia. Currently he manages a 25 strong programming group at the University of Western Australia and is secretary of the Western Australian branch of the ACS. He is active in the promotion of knowledge based computing techniques within the industry.
Book Reviews

CLEAVELAND, J. CRAIG (1986): An Introduction to Data Types, Addison-Wesley, 239 pp., $25.95 (paper).

This book can be strongly recommended to anyone with an interest in data types. Its thirteen chapters cover all the major issues, starting with a survey of the more familiar types and working through to abstraction, polymorphism and specification. The concepts are presented very clearly and are amply illustrated from a wide variety of languages. Each chapter concludes with a comprehensive bibliography and some well-chosen exercises.

The author adopts the conventional view that a data type is a set of values and a set of operations on those values. Other views receive only brief mention. The treatment is mostly non-mathematical. However, the last three chapters venture into theoretical territory and give a helpful introduction to a landscape which, for many of us, can appear forbidding. So, if you are curious about the claim that data types are homomorphisms of many-sorted algebras, you may find special value in these pages.

One point that calls for comment is the neglect of error conditions. In Chapter 8, for example, the author presents a series of modules for defining and implementing bags, but never addresses the problem of insertions into bags that are full. These have a null effect, with no suggestion from the programs that anything might be wrong! Languages such as C and Ada allow exceptions to be included in the type specifications, and a treatment along these lines would have been much better.

In every other respect the book is an excellent and very welcome addition to the literature on data types.

J.B. Hext
Macquarie University


"This book provides a detailed understanding of DL/1 data bases and how they perform."

Part I introduces DL/1 data bases.

Part II presents performance aspects of DL/1 access methods including HSAM, HISAM, HDAM and HIDAM and defines, in a decision table, criteria for the selection of an access method for a given application.

Part III describes performance aspects of physical design options such as block sizes, grouping of data sets (usually so that segments of similar length reside in one file), the use of secondary indexes, the incorporation of pointers and the exploitation of 'logical relations' to reduce physical storage requirements. The part includes some qualitative physical design guidelines.

Part IV discusses the effects of several aspects of the system environment, including the effect of the number and size of buffers in the pool, the frequency of synchronisation points and the pattern of calls from the application program to the DBMS, upon performance.

Part V gives space and time calculations which allow the prediction of data-base performance. Finally, there is a brief chapter which gives guidance on how to model actual program performance so as to obtain more accurate performance estimates than those obtained from rule of thumb calculations.

I have to admit that I am not immediately familiar with the IMS (DL/1) environment, but I have taught courses in Quantitative File Design for the last 20 years. Thus I am certain senior programmers and data base managers practising in the DL/1 environment will find this book a treasure and wonder how they have survived so long without it!

This book will allow the design, optimising and tuning of DL/1 data base programs to obtain greater throughput and faster response times for a wide variety of applications.

A.Y. Montgomery
Royal Melbourne Institute of Technology


This is a handy pocket reference book for Cobol 85 programmers. It makes no attempt to teach. It is organised like many other Cobol references, with chapters on the meta-language, overall program structure, the Identification, Environment, Data and Procedure Divisions, and language concepts. There is an appendix listing reserved words, a second appendix listing the features not covered, and a five-page index.

A pocket-size Cobol reference has to leave something out. Most of what is missing won't be missed. For example, where the standard has two pages of restrictions on the use of PERFORM, this book simply states, 'recursion is not permitted'. It may be an over-simplification, but for most programmers it is close enough. Occasionally the account is misleading. The description of UNSTRING with the POINTER clause is wrong, because it does not discuss the effect of the initial value of the pointer. However, this was the only error I spotted. In almost all cases the description of each Cobol feature is concise, readable, and accurate.

For the working Cobol programmer who is not a weight-lifter, this book should prove to be much lighter reading than an ordinary reference manual. It could also be useful to a tertiary student who already knows how to program, but wants a guide to Cobol syntax.

Depending on the reader, a weakness of the book may be the choice of topics that it has decided not to cover. Many of these omissions are obsolete features of Cobol that should not be used in new programs. I have no complaint about those. However, the Report Writer and Communications modules are not obsolete features, and the first, at least, is widely used.

Also, amazingly, the use of qualified names has been omitted. In Cobol, if I want to refer to component 'A' of record 'B', I write 'A of B'. Cobol also lets me write 'A' on its own, provided there is only one 'A' in the program. Some programmers find writing 'of' so tedious, that they make every name in a program unique, even if they have to spell 'Account-Number' as 'Account-no' at some times, and 'Acct-no' at others. I don't see why this book has to encourage them. It is not the place of a summary to censor the standard in this arbitrary way. Given that the book has only 147 pages, I don't see why a few more pages couldn't have dealt with these important topics. It would still be a slim volume.

The value of this book to you is therefore going to depend on whether you like your Cobol programs to be readable, and if you will ever want to use Cobol to create printed reports.

Barry Dwyer
University of Adelaide


When we began teaching Prolog at the University of New South Wales around 1980, the literature consisted almost entirely of internal reports from various University groups and relatively inaccessible conference papers. Over the past three or four years, that situation has changed dramatically and there is now a number of books for readers at all levels from beginners to experienced programmers. Malpas' book is directed at the latter category.

It begins with a 'Chapter O' which gives a brief overview of the logical basis for Prolog — from Aristotle to automatic theorem proving in a little under forty pages and then on to the development of Prolog itself. Whether this chapter should be read before or after the rest of the book depends very much on the reader's background. This material is certainly not a prerequisite to learning the language but does sometimes help in understanding why programs behave the way they do — and it is an interesting subject in itself.

Chapters 1, 2 and 3 introduce the main features of the Prolog language. The presentation is more concise than that in introductory texts.
but is probably paced at an appropriate level for the intended readership. Because logic programming is quite different in concept from programming in procedural languages like C, Pascal or Fortran, it is important that the details of syntax, etc which make up the body of a language like Prolog are imbued with the appropriate spirit. Chapter 4 discusses the important subject of ‘Approaches to Prolog Programming’. With the introductions out of the way, Chapters 5, 6 and 7 go on to look, in detail, at specific areas of application:

- Text processing (lexical analysis, parsing, a DEC-10 to micro-Prolog translator and a data-base query language).
- Knowledge representation (semantic networks, frames, object-oriented programming and inheritance mechanisms).
- Expert consultation systems.

The author has been careful to ensure that examples throughout the book conform to ‘core Prolog’. But he has also performed a very useful service by giving, in an appendix, detailed comments on the syntax, development environment and other features of seven of the most widely used flavours of Prolog (CProgol, Quintus, Slogic, Prolog-2, Arity, UNSW and Turbo). Other appendices give a summary of basic terminology, a collection of utility programs, answers to exercises and most importantly, a case study of a ‘real-life’ scheduling program for a manufacturing plant.

In summary, this book brings together the logical foundations of Prolog, a tutorial on the language itself and a range of examples illustrating more advanced programming techniques. It is probably aimed slightly below the level of Sterling and Shapiro (1986), and its presentation is more informal and less cohesively structured. It should, nevertheless, make worthwhile reading for those wanting to go beyond the basic introduction to Prolog.

References

R.A. Sammut
Australian Defence Force Academy


Are you interested in the design of new programming languages? Then this book may be for you.

The author’s main premise is that the sole measure of a programming language is its expressivity, and that to this end, all the abstractions dealt with by a language should be ‘First Class Citizens’. That is, that types, data items, processes, executable ‘code’ items etc., all can be manipulated in much the same ways e.g., passed as parameters, bound to names in an environment, applied, evaluated, printed and so on. This means that abstractions can be built to provide exactly what the users need, not what the language designer thought they would need. He suggests that polymorphic languages, which change the function of operators and routines according to the types of the arguments (like the ‘overloaded’ operators of C++), are more appropriate than non-polymorphic languages for this purpose.

To support these claims, he goes on to describe a system in which this is true. Certainly, within the system described, it appears easy to build up complicated structures, many of which are awkward to express in most computer languages. For example, the *for* construct found in most languages as a primitive for iteration where the item sequenced over is an array is here built up from other constructs in a general way, so that linked lists, matrices, strings, and code (amongst others) can be sequenced over using the same semantics and syntax in each case.

To give an example of the interesting style and enthusiasm with which the book is written, here is a portion of the first paragraph:

We think that the world exists because we perceive it; but do we all perceive it in the same way? If so, then given that that which we perceive is not actually the world, but is a conceptual model, does this exist, or is it just in our mind?

From here, the author goes on to talk about the problems of modelling the real world in a computer, and puts forward his argument for expressivity.

Unfortunately, the programming language used in all the examples is somewhat unusual, and nowhere is given a description of its syntax, nor a list of operations and types which may be considered primitive. For example, whereas in Pascal, Modula and related languages, the dot is an operator (for dereferencing fields of a record, or entry points of a module), in this language the dot is a letter, and is used merely to punctuate names of symbols. At first, this was quite confusing.

Moreover, many times the examples rely on things which have not been introduced at the point of the example. As understanding the examples is key to understanding the book, this means that the book cannot be understood on one reading. Typographic errors are also frequent.

Despite its drawbacks, the book is worth reading twice. The ideas expressed are very interesting, and, apart from the obvious errors, put well. The author finishes with a challenge:

The challenge is to design a machine architecture which efficiently implements dynamically typed languages. Interestingly, it appears that at least one manufacturer has taken up his challenge and is building a machine, the Rekursiv, which attempts to do this.

Peter Chubb
University of New South Wales


This book is the proceedings of the Second Conference on Modelling Techniques and Tools for Performance Analysis, held in Sophia Antipolis, France, in June 1985. (Proceedings of the previous conference were reviewed in the ACJ, Vol. 18, No. 4, 1986, p. 203). These conferences are organised by AFCET with the support of the IEEE Computer Society, ACM, University of Paris VI, INRIA and CNET.

The book contains 20 regular conference papers, of which two are in French, the rest in English. The first chapter was intended to contain an additional seven papers by invited experts, but two of them submitted only abstracts (one of two sentences in length). The papers submitted are:

- ‘Workload modelling techniques’ by G. Serazzi.
- ‘Experience with tools for software performance engineering’ by C.U. Smith (a valuable discussion on automated tools to support software performance engineering but illegible figure).
- ‘Dependability evaluation of fault-tolerant multiple processor systems’ by K.S. Trivedi et al. (a general overview of the major problems inherent in modelling of fault-tolerant multiprocessor systems; dependability of a system here means its reliability, accessibility, or a combination of both).
- ‘Design of modelling tools for computer capacity planning’ by J.P. Buzen (a short essay on commercial software packages based on the ideas of queuing networks for computer capacity planning).
- ‘The role of software reliability measurement in information systems’ by J.D. Musa (a reprint of a paper from ‘GlobeCom’84 Conference Record’; a presentation of two mathematical models of software reliability).

Other papers, divided (sometimes inexplicably) into seven chapters (‘sessions’) on modelling tools, case studies and local networks, represent a wide spectrum of topics and levels of discussion, from papers written by theoreticians for theoreticians, to papers written by practitioners (or at least for practitioners). Among these theoretical ones, readers will find an analysis of an M/G/1 queuing system (‘An M/G/1 queue with variable vacations’ by B.T. Doshi, p. 67) and of expected delays in open queuing networks under priority service scheduling (‘Modelling priority queuing characteristics in approximate analytical tools for open queuing networks’ by D.A. Stanford, p. 131).

There is a series of papers on modelling and performance evaluation of data communication networks, such as:

- ‘Transient analysis of random access method via diffusion approximation’ by H. Miyahara et al., p. 83 (the distribution of the number of backlog packets under Carrier Sense Multiple Access protocol with Collision Detection has been determined).
Book Reviews

' A proof of channel stability of D-ary symmetrical tree algorithms' by S. Muro et al., p. 343 (an analysis of collision resolution protocols for random access communication channels).

Strategies for communication with remote data bases via a network are analysed by K.S. Natarajan in 'Performance analysis of prephch strategies for database access', p. 207. A modified model for fault-tolerant systems is presented by H.M. Khelalfa et al. in their paper titled 'Degradable computer systems: A new approach to modelling and design optimisation', p. 295.

Among other papers readers will find reports on software tools for performance analysis, such as:

- 'SPADE: A tool for performance and reliability evaluation' by R.A. Sahner and K.S. Trivedi, p. 147 (a tool for statistical analysis of directed acyclic graphs that can be applied for the evaluation of concurrent program execution time and in analysis of fault-tolerant systems).

- 'A software tool for the workload analysis' by M. Calzarossa and G. Serazzi, p. 165.


An application of Petri nets is presented in a paper titled 'Construction of generalised stochastic Petri net models of bus oriented multiprocessor systems by stepwise refinements: A case study', by M. Marsan and G. Chiola, p. 259.

There is also a paper on 'Performance measurement of concurrency control methods in distributed database systems' by S.C. Moon and G.G. Belford, and on database management systems ('Performance administration using an expert system approach' by C. Frasson, p. 191), as well as on performance evaluation of computer systems ('Experience with a system to evaluate the performance of a time sharing computer system' by N. Fujimura and K. Ushijima, p. 291).

Because of diversity of problems considered it would be difficult probably to find a person interested in all topics discussed in this proceedings, but generally good level of presentations makes this book a valuable possession on shelves of institutional and university libraries.

Krzysztof Pawlikowski
University of Canterbury, New Zealand


The first edition of the book was published in 1983, before the GKS Standard (ISO 7942) was published in July 1985. This edition is a revision of the earlier material, modified to conform to the GKS Standard.

The book provides an easy-to-read introduction to the graphics function in GKS. Part 1 begins with a review of the evolution of the Standard and covers the important concepts of GKS in a logical sequence (graphical output, co-ordinate systems, segments, graphical input devices, styles of interaction and workstations). Part 2 addresses the more advanced topics such as GKS environments, control of input devices, segment storage, metafiles and attributes of primitives. For my interests however, there is not enough detail on some of the latter topics; for example, 'cell array', the 'generalised drawing primitive' and the 'escape' functions are treated in a brief five pages in Chapter 12.

The material is presented with numerous examples, which are expressed in a dialect of FORTRAN 77 for readability. The names used for the GKS functions are mapped into the FORTRAN equivalents which follow the now draft international standard for the GKS FORTRAN language binding (ISO DIS 8651, September 1986). This allows the examples to be easily related to FORTRAN implementations of GKS.

I found some of the conventions of the presentation rather irritating. For instance, examples are usually kept on a single page resulting in too much white space, references to GKS functions are not indented or highlighted and the five graphic input classes are presented in different orders in three chapters. Another disappointing feature is that most of the examples are kept on pages and therefore programming the examples wouldn't be very meaningful for many applications.

Nevertheless the book would serve as an excellent text for an introductory graphics course covering GKS. It would also be extremely useful for applications programmers who are considering using (or about to use) a GKS package, although I would expect that the book wouldn't be used as a reference after some experience with the package.

John F. O'Callaghan
CSIRO Division of Information Technology


Understanding Statistics comprises 326 pages of an elementary statistics text, 89 pages of manual for the Statpal package, and a 5½" IBM-PC floppy disk containing the Statpal package (v.5.0) and two sample data sets.

The text itself is an introductory course in statistical methods, suitable for students with very limited mathematical background (certainly pre-calculus). The author makes a point of explaining concepts and methodology rather than producing a recipe book, and makes a very good job of it. The contents list is standard: sampling, inference, 'describing data' (point estimates), normal and binomial distributions, interval estimation, hypothesis testing, inference on means, proportions, and differences of means, one and two-way analysis of variance, correlation and regression, categorical data. Nonparametric tests are included in the appropriate sections. There is very little coverage of graphical techniques for exploratory data-analysis. Two appendices give the data sets also included on the Statpal disk. The text itself stands independently, although many of the exercises refer to the Statpal package.

Statpal claims to be a general-purpose package, 'very easy to learn and use, yet provides the power and sophistication needed by professionals'. The first of these claims is certainly true, the second not so. Statpal is completely menu driven, but not too irritating for that. It provides convenient facilities for editing data and adding new cases and variables, although the format of data is quite limited. Tools provided include scatterplots, histograms, breakdown, crosstabulation, nonparametric tests, correlation, regression and analysis of variance (simple k-way designs only). A simple programming language allows transformations of data such as LOGWT -- log(WT), but cannot do DEVWT -- WT -- mean(WT) for example, and has no flow control apart from an IF statement. I cannot vouch for the numerical stability of the package as no benchmark data sets are included, but the manual does at least give references for its numerical methods.

I would not use Statpal as a research tool — what it does, it does well but I could never be sure, with any data set, that I wouldn't eventually want to do more than Statpal is capable of (looking at contrasts in an analysis of variance for example). The main use of Statpal must be as a teaching aid for an introductory course in statistics, and as a simple tool for statistical packages to provide some demonstrations. The description is sensible and exaggerated only. The two form a commendable partnership at a reasonable price. Even so, I would ponder on whether the time spent introducing Statpal might not be better spent on a less friendly but more powerful package such as SPSS or SAS.

E.A. Catchpole
Australian Defence Force Academy


Judging by the price and the small number of pages one might expect of this book a very technical text directed at IBM PC AT specialists. Or perhaps less technical insights into the details of the IBM PC AT a la Peter Norton's books, such as Inside the IBM PC AT. However, both the technical and non-technical readers will be disappointed.

The first quarter of the book (or 35 pages) is taken up by a, for IBM, very flattering history of IBM followed by a chapter on IBM PC AT hardware/software configurations with many boring details of the AT, such as weight, cable lengths, dimensions, humidity ranges, and BTU output. The technical details appear to have been copied straight from IBM brochures without regard to the intended non-technical audience or relevance. Terms, such as 7-channel DMA or 16-level interrupt are

The Australian Computer Journal, Vol. 19, No. 4, November 1987
Book Reviews

presented under the heading of AT highlights and will mean little to the reader. The same problem exists on the software front, e.g., take the highlights of the IBM PC XENIX Software Development and Text Formatting systems. Non-technical readers are bombarded with XENIX features, such as the lexical analyzer ('lex') and compiler-compiler ('yacc') and technical readers (including myself) are left wondering what sort of goodies the Berkeley extensions 'strings', 'xstr', 'xalloc', and the 'explain', and 'style' hold in store. Did you know that the IBM PC XENIX General Information Manual contains 24 pages in 8.5 by 11 inch format? These two irrelevant facts are considered to be highlights of the manual and appear under the separate bold heading of highlights!

The third chapter contains IBM PC AT speed tests; the comparisons are made between the IBM PC and the IBM PC AT for short BASIC and C programs. In average, the IBM PC AT is shown to be faster than the IBM PC. Various IBM PC AT alternatives, such as the Sperry PC/IT, Texas Instrument's Business Pro system, the COMPQA range, and the NCR PC8 are discussed in the fourth chapter.

The largest chapter with about 50 pages (or one third of the book) is taken up by a simple introduction to programming in PC-BASIC and includes many short sample programs. As a lot of the other AT material can be gleaned from glossy brochures, I believe the last chapter on BASIC programming represents the most useful part of the book but it is doubtful whether many intending PC AT users will spend $37 for 50 pages of introductory PC-BASIC programming plus 35 pages of IBM's history.

Dominic Wild
Perth Technical College


This book impressed me as an excellent introductory text on database design principles and techniques, with emphasis on 'introductory'. This is perhaps the first text in a difficult field of data-base design, which does not require any special background in databases, other than a vague idea of what database systems are about. For a specialist, the book does not provide much new except in the way of presentation, which is 'reader-friendly' with extensive use of self-explanatory pictures and wonderful editorial work (in either respect, it compares favourably with an earlier contribution by the same author, Vetter and Maddison, 1981).

The reviewed book may be compared not only with previous Vetter's work but also with other books in this increasingly competitive market. Current competition includes Furtado and Neuhold (1985), Hawryszkiewycz (1984), Howe (1983), Yao (1985), Teorey and Fry (1982) and Vesely (1986). Furtado and Neuhold's monograph presents in a formal way basic techniques for conceptual and logical design. Hawryszkiewycz's book is a comprehensive text, whose major strength is in conceptual design and relational design methods (where the author's expertise has been widely acclaimed). This reviewer is using Hawryszkiewycz's book as a textbook course in databases. The book by Howe is interesting, but not very well-structured or consistent reading. Conceptualisation and, to a lesser extent, logical design are covered satisfactorily. The book edited by Yao is not really a text, but a collection of articles. Admittedly, an editorial touch is easily sensed and the articles are well placed in the overall structure (conceptual and logical designs are covered). The text by Teorey and Fry has become slightly dated in that it is about the design of file, rather than database, structure. Otherwise, a very good book, Vesely's book promises more than it delivers. Its presentation is neither complete nor systematic. In my opinion, the reviewed book stands comparison with all these books.

This said, there are aspects of Vetter's text about which this reviewer is quite critical. They relate not to the presentation, however, but to the chosen methodology, which is limited to the design of data structures. The design of processes is not only beyond the scope of the book but the influence of processes (i.e. identified business transactions and events) on data structures is not felt (except for one short section in the chapter on logical design). An immediate consequence is that the methodology is data-driven. A further consequence is that the conceptual design is done in terms of well known relational database design strategies — synthesis and decomposition. This comes as no surprise, since the relational strategies are data-driven. The relational definitions are derived from the nature of the data, expressed in terms of various dependencies, integrity constraints, and inference rules. Such designs are quite insensitive to the dominant user requirements on data and tend not to be directly tuned to the current (and desirable) pattern of business functions. Vetter's data-driven strategy goes as far as designing a conceptual entity-relationship structure in terms of relations. Such a strategy, although not new, is one which stands to the conceptual design principle, i.e. to be independent from any computer-based model. It obscures the difference between the conceptual and logical designs, and, if the implementation system is relational, both designs are in fact identical. This, in turn, leads in the book to the presentation of the physical design partly in terms of the logical data structures (such as Codasyl records and sets). Indeed, many typically physical structures are not addressed (page, storage record, area, partition, file, block, etc.).

The final chapter of the book promises to discuss the data dictionary as a design tool. This promise is not fulfilled. The data dictionary, to serve as a design vehicle, must define the data structuring primitives of various database design phases (entities, relationships, attributes, tables, columns, domains, Codasyl sets, business functions, etc.). Clearly, this is a higher dictionary layer than the definition of a meta-database intension, as discussed by Vetter. In other words, Vetter describes a typical data dictionary system as opposed to a design data dictionary to support automated design tools. In this circumstance, the author's claim that his methodology 'can be used — if desired — to generate data structures automatically' is not substantiated. In my view, the book would benefit from dropping the data dictionary chapter altogether.

The above criticisms do not seriously undermine the book's strengths. They may well be interpreted as mirroring the reviewer's prejudices, and the author's inability to explain fully his standpoints in an introductory text. This reviewer recommends the book highly to students and teachers of database courses, mostly for the way in which the material is presented. For those with only a limited grasp of database technology and looking for a truly introductory text on database design, the book may be invaluable. Database processing professionals can draw inspiration from Vetter's systematic and elegant approach. For all those, buying the book can mean money well spent.

References

Leszek A. Maciaszek
University of Wollongong


In recent years there has been progress in understanding how collective properties of various kinds emerge from systems of interacting elements or cells. The work of John Hopfield on neural networks provides a good example (Hopfield, 1982; Ferry, 1987). Assuming binary state neurons and symmetric interactions, Hopfield was able to demonstrate the convergence of such systems to stable states of neuronal activity. Since the stable states, interpreted as memories, could be selected in advance they may well be interpreted as mirroring the reviewer's prejudices, and the author's inability to explain fully his standpoints in an introductory text. This reviewer recommends the book highly to students and teachers of database courses, mostly for the way in which the material is presented. For those with only a limited grasp of database technology and looking for a truly introductory text on database design, the book may be invaluable. Database processing professionals can draw inspiration from Vetter's systematic and elegant approach. For all those, buying the book can mean money well spent.
evolution, molecular evolution, and cellular growth. Many of the papers in the present book relate to these programs.

The papers, thirty-six in number, were presented at a meeting of the French Mathematical Society held in late 1983. They fall into five sections:

1. Introduction and Review (Demongeot, et al).
2. Dynamics of Cellular Automata: Theoretical Approach (Fogelman-Soulie).
3. Applications to Growth Models (Wolfram).
5. Applications to Computer Science (Kung).

The Introduction and Review does not provide much guidance as to key papers. The bracketed authors are those I have judged to be key contributors based on the quality of their contribution or frequency of citation by other contributors.

Section 4 is the major section in terms of number of papers (eleven) and, in my opinion, the most interesting. Kauffman’s paper ‘Selective Adaptation and its Limits in Automata and Evolution’ is concerned with how the theory of automata throws light on the mechanism of species evolution. Kauffman defines three classes of automata by type of interconnection: completely connected, threshold (cf. Hopfield), and canalizing. In canalizing automata elements are interconnected by boolean input-output functions that are assigned randomly at initialization time. Boolean functions such as OR introduce so called ‘forcing’ connections which propagate element values and so reinforce patterns of activity. The canalizing automata converge to asymptotic states that exhibit interesting structure from the point of view of modelling gene interactions. For example, functionally isolated subsystems of elements (genes) are found and these subsystems can be independently modified by mutation so allowing for piecemeal improvement in behaviour. By contrast, in fully connected automata, modifications such as element loss preclude gradual improvement. This provides a way of linking global properties of genetic automata (genomes) with fitness. Weisbuch in ‘Modelling Natural Systems with Networks of Automata: The Search for Generic Behaviours’ explores this connection. Starting with equations describing the dynamics of population of genomes he is able, with the aid of computer simulation, to show convergence to quasi-stable states (species) followed by periods of rapid transition. This provides an understanding of the phenomenon of ‘punctuated equilibrium’ observed in the paleontological record. On a different tack Allan in ‘Self Organisation in Probabilistic Automata Networks’ shows how canalizing networks subjected to perturbation can exhibit computational properties such as the ability to recognise patterns.

Except, possibly, for the key papers I found papers in the other sections pretty esoteric and these sections left me a bit cold. Fogelman-Soulie’s paper ‘Stable Core in Discrete Iterations’ provided some underpinning for the results assumed above about canalizing networks. Wolfram’s paper ‘Some Recent Results and Questions About Cellular Automata’ was basically a review of growth models. Those familiar with ‘fractals’ may find this paper interesting. Most of the papers in Section 5, including Kung’s ‘Two-Level Pipelined Systolic Arrays for Matrix Multiplication, Polynomial Evaluation, and Discrete Fourier Transform’, were devoted to systolic array processors, i.e. special purpose VLSI architectures involving pipelining and multiprocessing. In my opinion these papers did not sit well with other papers in the volume.

I find it hard to get enthusiastic about this book. On the whole the papers are approachable, except possibly for some of the more mathematical ones in the theory section. The key papers repay rereading. I would not recommend the book to novices wanting an ‘inside’ view of the field. Researchers would get a snapshot of activity but one that is now a bit out of date. Some papers suggest lines for further enquiry, e.g. Kauffman’s. The book is cheap enough for most universities to have a copy in their main library.

References


Brian Lederer
CSIRO

The Australian Computer Journal, Vol. 19, No. 4, November 1987


Despite its general title, this book describes the design, operation and application of a parallel image processing system, developed in the Department of Physics and Astronomy, University of York, London. The initial versions of the system, called the Cellular Logic Image Processor (CLIP) were developed in the early 1970’s. The latest processor CLIP4 was built in 1980 and a commercial version of this processor was released in 1984. The difficult path to commercialising this kind of technology is outlined in the Introduction to the book.

The Chapters consist of parts of Theses from eight PhD students working on the CLIP4 project. Chapters 1, 2 and 3 describe the software environment and functions of CLIP4, while Chapters 4 to 8 describe the implementation of image processing algorithms for several realistic applications — serial reconstruction, colony counting, tomography, motion analysis and automatic segmentation. The final chapter outlines future developments of the CLIP (CLIP7) to handle large images and to provide different control for mechanisms for multiprocessor configurations.

A brief introduction to each chapter provides a context for the material, which is well-written and easy to follow. The book therefore provides an excellent introduction to the development of image processing algorithms on parallel computer architectures.

The value of the material is limited however by the restrictions in the design of CLIP (e.g. its lack of floating point operations and colour image processing functions) and the lack of comparison of the CLIP design with other commercial systems (e.g. ICL’s DAP, NASA’s MPP, Lucasfilm’s PIXAR) and with other prototype systems (including for example INMOS Transputers and data-flow processors). Unfortunately most of the material in the book was written before 1985 (the date of the most recent reference is 1984) and therefore readers will not find information on the recent developments in multi-processor configurations for image processing.

John F. O’Callaghan
CSIRO Division of Information Technology


In a fast-moving field such as computer engineering, by the time a technology is understood sufficiently well to design a course around it or write a textbook on it, it is out of date. The only recourse appears to be to teach basic principles (which tend to change slowly over time) using the most up-to-date examples available. This is what Professor Wilcox has done in this book.

Using the design and implementation of an S100 bus 68000-based microprocessor board as its central example, the author describes the course in engineering design methodology. Defining the problem, exploring solutions (analysis), synthesising a workable solution, designing the implementation at a high level, evaluating the result and iterating — it’s all here, including the very important topics of documentation and review. Of course, the S100 bus is now almost obsolete — one thinks of it in terms of 8080s and CP/M — but this does not detract from the author’s purpose in teaching basic Engineering Design. The requirements of sound design are the same, no matter what microprocessor and bus are used — even if the project is totally unrelated to microprocessors.

A student doing a course such as the one described in the teaching notes given in an appendix would, by the time the course was complete, have constructed a working microprocessor prototype, and have documented it fully. He would have learnt a design technique that is quite general, and a number of details which can only be learnt by doing a design. However, the course as taught at Bucknell University in the USA, was fairly time-intensive, with students putting in an average of twenty hours a week.

Each section has a set of exercises, and worked examples are scattered liberally throughout the text. Three example reports are included in the appendices, one of them written by a student who has done the course.

I imagine that an Engineering course based around this text would, by the time the course was complete, have constructed a working microprocessor prototype, and have documented it fully. He would have learnt a design technique that is quite general, and a number of details which can only be learnt by doing a design. However, the course as taught at Bucknell University in the USA, was fairly time-intensive, with students putting in an average of twenty hours a week.

Each section has a set of exercises, and worked examples are scattered liberally throughout the text. Three example reports are included in the appendices, one of them written by a student who has done the course.

I imagine that an Engineering course based around this text would, by the time the course was complete, have constructed a working microprocessor prototype, and have documented it fully. He would have learnt a design technique that is quite general, and a number of details which can only be learnt by doing a design. However, the course as taught at Bucknell University in the USA, was fairly time-intensive, with students putting in an average of twenty hours a week.

Each section has a set of exercises, and worked examples are scattered liberally throughout the text. Three example reports are included in the appendices, one of them written by a student who has done the course.

I imagine that an Engineering course based around this text would, by the time the course was complete, have constructed a working microprocessor prototype, and have documented it fully. He would have learnt a design technique that is quite general, and a number of details which can only be learnt by doing a design. However, the course as taught at Bucknell University in the USA, was fairly time-intensive, with students putting in an average of twenty hours a week.

Each section has a set of exercises, and worked examples are scattered liberally throughout the text. Three example reports are included in the appendices, one of them written by a student who has done the course.
The Australian Computer Journal, Vol. 19, No. 4, November 1987

Book Reviews

Peter Chubb
University of New South Wales


Over the last few years Complementary Metal-Oxide-Silicon (CMOS) technology has become more and more popular for the fabrication of Very Large Scale Integration (VLSI) Integrated Circuity (ICs). This book is a collection of articles by thirteen authors (including the editor) involved with CMOS technology in the microelectronics industry largely in the United Kingdom with one from Belgium and one from the United States. Most of the authors are from large companies such as Texas Instruments and Plessey and others are from educational institutions. Choosing and Using CMOS is aimed largely at design engineers wanting to know how CMOS can solve modern circuit fabrication problems. Very little material in this book has been written for the software fraternity although a few interesting references to Silicon Compilers and well known SPICE analogue simulation package are made.

Most of the articles deal with digital CMOS circuitry although there is one by Dr John Pennock of The Wolfson Microelectronics Institute in Edinburgh that deals with analogue techniques. And apart from two or three articles on standard CMOS logic and design techniques, the bulk of the book seems largely aimed at the custom IC design market.

A chapter by Bob Whelan of Array Logic describes the Gate Array, a prefabricated custom CMOS IC in which the hardware designer simply designs the connections between the various transistors that make up each logic gate and the vendor then fabricates these connections onto the premanufactured wafers. Aspects including testability, design and simulation are covered amongst others.

R. Heaton of Acorn Computers Ltd of Cambridge, a well-known personal computer manufacturer, follows on with another discussion of Standard Cell Approaches to Custom IC Design in which he looks at the relative merits of various design customisation techniques including PROMs, PLAs and semicustom ICs. He follows on with a look at development procedures and schedules and finishes with a look at future developments.

Possibly the most interesting chapters as far as a design engineer is concerned are: Casebook design of a Gate Array by Chuck Pettit of Gothic-Cre II I on Limited and Casebook design of a Custom Integrated Circuit by Philip Woodhead of Hughes Microelectronics Limited in which Custom ICs are designed using the PC00450 gate array from Philips and the HML975 full custom IC designed by Hughes for a European supplier. Many aspects of the procedure involved in designing custom CMOS ICs are covered including which software packages to use and the respective machines to run them, time requirements and batch sizes. These two chapters should prove quite informative to a design engineer.

The last chapter by Alan F. Murray of the University of Edinburgh discusses the future of CMOS technology especially in respect to VLSI circuitry. This book will find its main audience with hardware design engineers looking for new ways of building complex circuitry more economically and expeditiously.

Robert Edwards
University of New South Wales


Books seem to come to me for review at very opportune times. I have recently had doubts about the effectiveness of our testing strategies. Will this book lead us to something better? In general, I think yes. If nothing else the bibliography led me to some interesting parts of the library shelves that I had not previously visited. Actually, more than this, it has provided a convenient synopsis of testing strategies and techniques for the entire cycle of software development.

It is a rather slim volume put together by a Working Group of the British Computer Society. The intention is to introduce the reader to the possibilities for testing at every stage of development — not just testing of code.

I think it succeeds very well. Each stage of development is briefly defined and then possibilities for testing succinctly described. Included in most sections are checklists that I think I will find of very practical assistance. As befits such a guide as this, the bibliography is quite good and refers the reader to more detail. Most of the reference material is quite recent — covering 1977 to 1986. It is well written with excellent cross-referencing. In some ways the material is a bit repetitious — but this is appropriate as many testing techniques are applicable to more than one stage of development.

The section on programmer’s testing was clearly focused on procedural or algorithmic programming languages. What about testing non-procedural languages and 4GLs? Perhaps these should be considered as being similar to formal design specifications.

I think it is a very useful guide for software developers and certainly worth purchasing for the library.

Randall Fletcher
Brisbane College of Advanced Education

Corrigendum


In Barry Dwyer’s review of this book published in this journal in August 1987 (Volume 19, Number 3, page 175), the word ‘not’ was inadvertently omitted from the sentence (column two, line 17):

It is worth noting that Jackson is not one of those writers who believes that a single technique can solve all problems.

The omission unfortunately reversed the reviewer’s intended meaning, and we sincerely apologise for any inconvenience or confusion engendered.

Addendum


This book was reviewed in this journal in May (Volume 19, Number 2, page 109). We have been advised of the availability of a low-priced student edition (details as above) that may be obtained direct from the publisher.

John Lions
Book Reviews Editor

The Australian Computer Journal, Vol. 19, No. 4, November 1987
WANG MANUFACTURING IN CANBERRA

Wang Computer's factory in Canberra is currently producing Wang 4245 colour workstations and 4230A monochrome workstations for export to Europe and the United States.

Wang Australia is the world-wide sole supplier of the 4245 with production running at a steady one hundred units per week from its manufacturing plant in Canberra's northern suburb of Bruce.

---

ELECTION OF FELLOWS COMMITTEE

Elections have been held to fill vacancies on the Fellows Committee for 1987-88 resulting in the appointment of the following Committee Members:

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ms P.M. Alexander</td>
<td>Chairman</td>
<td></td>
</tr>
<tr>
<td>Mr F.L. Hall</td>
<td></td>
<td>New South Wales</td>
</tr>
<tr>
<td>Dr D.F. Bright</td>
<td></td>
<td>New South Wales</td>
</tr>
<tr>
<td>Mr. A.W. Goldsworthy</td>
<td></td>
<td>Queensland</td>
</tr>
<tr>
<td>Professor R.S. Northcote</td>
<td></td>
<td>South Australia</td>
</tr>
<tr>
<td>Mr A.H. sale</td>
<td></td>
<td>Tasmania</td>
</tr>
<tr>
<td>Mr P.M. Murton</td>
<td></td>
<td>Victoria</td>
</tr>
<tr>
<td>Mr T.A. Reid</td>
<td></td>
<td>Western Australia</td>
</tr>
</tbody>
</table>

---

APPLICATIONS ARE INVITED FOR APPOINTMENT TO THE POSITION OF POSTDOCTORAL FELLOW/RESEARCH FELLOW IN COMPUTER SCIENCES

DEPARTMENT OF ENGINEERING PHYSICS
RESEARCH SCHOOL OF PHYSICAL SCIENCES

Applications are sought for a full-time research position in the Computer Sciences Laboratory.

The Computer Sciences Laboratory studies (1) parallel algorithms and architectures, including systolic arrays and transputer networks, (2) both fundamental and applied aspects of human-machine systems, including image analysis, speech recognition and speaker characterisation. An interdisciplinary project involves information processing aspects of perception in biological systems.

Applicants with a strong background in any related area of Computer Science are encouraged to apply.

Further information may be obtained from Professor R.P. Brent.


Appointment: Research Fellow up to three years, possibility of extension to five years; Postdoctoral Fellow normally two years, possibility of extension to three years.

Applications should be submitted in duplicate to the Registrar, The Australian National University, GPO Box 4, Canberra ACT 2601, quoting reference number and including curriculum vitae, list of publications and names of at least three referees.

The University reserves the right not to make an appointment or to appoint by invitation at any time. Further information is available from the Registrar.

THE UNIVERSITY IS AN EQUAL OPPORTUNITY EMPLOYER

---

BINDERS FOR THE “AUSTRALIAN COMPUTER JOURNAL”

NOW YOU CAN STORE YOUR COPIES OF THE “AUSTRALIAN COMPUTER JOURNAL” CONVENIENTLY FOR EASY ACCESS AND REFERENCE

The “Australian Computer Journal” binder will hold twelve copies (3 years). Each magazine is held separately by a wire for ease of opening and removal.

Available in colour blue and clearly labelled on cover and spine, this binder will complement your library or office.

Price: $10.00 per unit, including postage and packing (add $5.00 for overseas orders).

Orders to:

ACS National Secretariat
PO Box 319, Darlinghurst, NSW, 2010

Please supply ............... ACJ Binders.

Payment of $A...................is enclosed.

Name .................................................................

Address ..............................................................

...............................................................................

Please allow three weeks for delivery within Australia.
CONTRIBUTIONS: All material for publication should be sent to Mr Rob Cook, Editor, Australian Computer Journal, Centre for Information Technology Research, University of Queensland, St. Lucia, Qld 4067. Prospective authors may wish to consult manuscript preparation guidelines published in the February 1986 issue. The paragraphs below briefly summarise the essential details.

Types of Material: Four regular categories of material are published: Papers, Short Communications, Letters to the Editor and Book Reviews. Generally speaking, a paper will discuss significant new results of computing research and development, or provide a comprehensive summary of existing computing knowledge with the aim of broadening the outlook of Journal readers, or describe important computing experience or insight. Short Communications are concise discussions of computing research or application. A letter to the Editor will briefly comment on material previously appearing in the Journal or discuss a computing topic of current interest. The term ‘computing’ is interpreted broadly to include descriptions of computer hardware and software, and major computer applications such as information systems, expert systems, computer networks and office automation.

Refereeing: Papers and Short Communications are accepted if recommended by anonymous referees. Letters are published at the discretion of the Editor, and Book Reviews are written at the Editor’s invitation upon receipt of review copies of published books. All accepted contributions may be subject to minor modifications to ensure uniformity of style. Referees may suggest major revisions to be performed by the author.

Proofs and Reprints: Page proofs of Papers and Short Communications are sent to the authors for correction prior to publication. Fifty copies of reprints will be supplied to authors without charge. Reprints of individual papers may be purchased from Associated Business Publications, PO Box 440, Broadway, NSW, 2007. Microfilm reprints are available from University Microfilms International, Ann Arbor/London.

Format: Papers, Short Communications and Book Reviews should be typed in double spacing on A4 size paper, with 2.5 cm margins on all four sides. The original, plus two clear bond-paper copies, should be submitted. References should be cited in standard journal form, and generally diagrams should be ink-drawn on tracing paper or board with stencil or Letraset lettering. Papers and Short Communications should have a brief Abstract, Key word list and CR categories on the leading page, with authors’ affiliations as a footnote. The authors of an accepted paper will be asked to supply a brief biographical note for publication with the paper.

This Journal is Abstracted or Reviewed by the following services:

Publisher: Service
ACM Bibliography and Subject Index of Current Computing Literature.
ACM Computing Reviews.
AMS Mathematical Reviews.
CSA Computer and Information Systems Abstracts.
Data Processing Digest.

ENGINEERING
INDEX INC. Engineering Index.
INSPEC Computer and Control Abstracts.
INSPEC Electrical and Electronic Abstracts.
ISI Current Contents/CompuMath.
ISI CompuMath Citation Index.
SPRINGER- Zentralblatt fur Mathematik und ihre Grengebiete.
VERLAG

Copyright © 1987. Australian Computer Society Inc.


All advertising enquiries should be referred to the above address.

Printed by MAPS Litho Pty Ltd, Unit 3, 8 Aquatic Drive, Frenchs Forest, NSW, 2086.