CONTENTS

TUTORIAL ARTICLE
70-76 A Tale of Two Professions — Civil Engineering and Computing
(1980 Hawken Address)
J.M. BENNETT

RESEARCH PAPER
77-82 On the Design of Process Assignee for Distributed Computing Systems
R.K. ARORA, S.P. RANA and N.K. SHARMA

REPORTS AND SURVEYS
83-86 Local Government Computing in the Geelong Region
R. PHILCOX
87-92 Attitudes Towards Computers: A Survey of Diverse Persons
A.G. SMITH

SHORT COMMUNICATIONS
93-94 ASYM: A Divisive Monothetic Classificatory Program for Binary Data
D.A. HEDGES and W.T. WILLIAMS
95-96 DOTX: A New Program for the Production of an Ordered Two-way Binary Table
H.J. CLAY and W.T. WILLIAMS
97-99 Remarks on "A Comparison between Pascal, Fortran and PL/I"
(with Author's Reply)
A.H.J. SALE and D.J. KEWLEY

SPECIAL FEATURES
69 Editorial
100-105 Book Reviews
106-107 Letters to the Editor
108 Call for Papers
"This wouldn't have happened, Fenwick, with a Tandem NonStop™ System."

When your computer's down, are you out of business?

You can bank on it. Timing couldn't be worse. Your busiest season. Customers pounding for service. All it takes is one small failure somewhere in the system. One disc or disc controller. One input/output channel. For want of an alternative, the system is down and business is lost. Sometimes forever. Unless you have a Tandem dispersed processing system.

NonStop™ Operation to be sure.
This is the only system you can buy which keeps on running, right through a failure that would shut down any other system on the market today. Including a failure in a disc, a disc controller, an input/output channel or one of the processors.
Tandem takes even a critical failure in stride.
No transaction-in-process is lost or duplicated.
No downtime.
No lost records.
No lost customers.

No question about it.
When you're on a Tandem NonStop™ System, your information files and your processes are protected from contamination in a way no other system can match. We've built in safeguards other suppliers can only dream about. You've heard the war stories about restart and restore data base on other systems. They're not exaggerations, but they can be a thing of the past. With a Tandem NonStop™ System.

Grow without penalty.
Start with only the computer system you need right now. You can grow it all the way from a local system to a global network. And never lose one cent on your original investment.

Same hardware. Same software. With low cost additions, not replacements. It's worth your interest.
News Briefs from the Computer World

"News Briefs from the Computer World" is a regular feature which covers local and overseas developments in the computer industry including new products, interesting techniques, newsworthy projects and other topical events of interest.

ACS NATIONAL CONFERENCE IN HOBART

The Australian Computer Society is to hold its national conference in Hobart next year.

The national conference, normally held every two years, was last held in Canberra in 1978. But the 1980 conference was suspended because the Eighth World Computer Congress was held that year with the joint venues of Tokyo and Melbourne.

The 1982 conference chairman, Mr Tony Haigh, of Hobart, said there had been tremendous development in technology during the past four years. This had led to the theme "native technology" being set, to give computer professionals, and all interested parties the chance to look at the direction of computer and computer-related fields in Australia.

"The intention is to examine the extent to which native technology can be encouraged to develop in smaller countries," Mr Haigh said.

"The building up of indigenous technology is of interest to Australia and surrounding countries, such as Singapore and New Zealand."

Mr Haigh said the main conference venues would be Australia's first legal casino, Wrest Point, and the University of Tasmania, both overlooking the panoramic harbour in Hobart, and a few minutes walk apart.

He said planning was already well in hand for the conference and the associated exhibition, planned to be one of the biggest in the southern hemisphere.

Warehouse on Hobart's historic Sullivans' Cove area had been booked for the exhibition, which could run, in terms of floor space, to more than 40,000 sq ft.

Mr Haigh said the conference would cover most relevant aspects of computing in Australia.

The organising committee hoped to be able to distribute the conference proceedings in advance on microfiche.

Because there had not been a conference for four years, the organising committee was expecting a large number of registrations, and asked people planning to attend to register early, Mr Haigh said.

GO FLY A PLANE!

Dick Smith Electronics have released a computer program for their Home Personal Computer, that teaches you to fly a plane (not a kite!) and you don't even have to leave your loungeroom.

The X3684 Flight Simulator Program from Dick Smith can teach you the laws and theory of flight in three dimension.

Full instrumentation is provided, and combined with "out-of-the-window" 3D dynamic flight display and radar, you can enter the world of aeronautics.

PLATO COMES TO DEAKIN

Victoria's Deakin University has become involved in research and development for the computer based teaching system, PLATO.

Deakin is the first university in Australia to take advantage of this unique instructional program.

Lecturers in Deakin's School of Education will be rewriting PLATO lessons for Australian students, adapting them to suit local curriculum requirements, and producing new courses in language and mathematics.

PLATO, an acronym for programmed logic for automated teaching operations, was developed at the University (Continued on Page III at the back)
He's very temperamental.

Because he's so touchy, you have to treat your computer with kid gloves to get the best out of him.

At Cemac Interiors we do everything from keeping him happy to spoiling him rotten.

Static gives him the hiccups:

Unsuitable floor coverings cause static charges that go to earth via a computer. And when he hiccups you get data loss or complete shut-down. To prevent this we install Cemac Tate anti-static carpet as part of our Cemac Tate Access Flooring System.

Access Flooring not only takes care of unsightly wires, cables and plumbing, but saves considerably on re-location time and expense. You have total flexibility with telephones, power outlets, lights, gas and water pipes and will save on central air conditioning costs.

Dust makes him sneeze:

Plaster and mineral fibre ceilings give off dust that gets into your computer and onto micro-chips, diodes, transistors and electrical connections. And when he sneezes it causes data loss and interruption, short circuits and machine shutdowns. So above their access floors Cemac Interiors install metal pan ceilings that don't make him sneeze.

Heat makes him blow his cool:

If a computer gets too hot it shuts itself off. So when he blows his cool, the result is very expensive downtime. That's why Cemac Interiors also offer single source responsibility for the installation of air conditioning and fire protection equipment as part of a total access flooring package.

Rip-offs leave him completely devastated:

If the confidential data he's racked his brain to produce falls into the wrong hands, he gets very upset. So Cemac Interiors arrange total security protection.

Cemac Interiors also supply ambient and task lighting, partitioning, and computer room work stations from the Westinghouse Open Plan Office System.

In fact everything to keep your computer working happily.

Give us a call and we'll put it all together for you.

Beautifully.

Cemac Interiors

Sydney 2903788
Melbourne 4196233
Brisbane 2215099
Adelaide 453656
Hobart 295444
Perth 4447888

CEIN 0057
Editorial

In view of the frequent complaint from ACS members about the Journal being too highbrow, it might interest them to know that criticisms from computing academics (including myself — before I became Editor) are just as unscathing. Whereas practitioner members find papers on specialised research topics irrelevant to their work, many academics object to the presence of elementary tutorial articles and industry case studies on the ground that they detract from the prestige of the Journal as a research publication. Speaking more generally, the controversy is merely part of the well-known gap between the two communities of Australian computer scientists and industry practitioners. While the chasm is bridged here and there by concerned groups and individuals, the overall tendency appears to be quite otherwise.

As a person who has substantially participated in the development of exclusively computer science activities, especially the annual Computer Science Conference separate from the ACS conferences, I am only too aware of the very real divergence of interests between the two sides, and for good reasons. It would be naive, however well intentioned, to expect the majority of people working in a commercial environment, engaged in developing and maintaining software products which are supposed to contribute to corporate profitability, to seriously share the values of scientists working in institutions where contributions in the form of printed words are the ultimate source of esteem and advancement. We can only expect that the two groups of people will go their individual ways: To each his favourite preoccupations and prejudices.

But the fact that two distinct communities exist and have separate basic concerns, need not preclude them from co-operation on specific matters; for example in case of the Journal, which is supported, financially and organisationally, by ACS, but also by academics in terms of contribution to contents. To be sure ACS derives a benefit from this involvement, in the sense that the Journal helps to establish the credentials of ACS as a learned society. Also, the history of the Journal over the past 15 years has been anything but peaceful. Still, it was because of its existence under the ACS umbrella, which provided the necessary subscription management facilities and low publications overheads due to its large membership circulation, that the Journal was able to gradually build up a substantial external circulation, such that it is now able to fulfill the desired function of bringing work done in Australia to the notice of an international audience. Having been involved with alternative publications for some time, I can speak with some assurance of the difficulty of attaining a similar state of affairs starting from scratch.

Thus, the Journal is the result of a longstanding but difficult partnership, and its contents must be looked at in this light. In view of the greater willingness of academics to contribute material to the Journal, it naturally occurs that the contents will reflect their interests. At the same time, since ACS members in industry pay the major part of the Journal’s publication cost, it is only fair that a special effort be made to obtain some industry-oriented material. The balance is not an easy one, but it need not be impossible. In short, I do not exhort practitioners and academics to love each other; merely to work together where such co-operation is of mutual benefit. Nor do I expect everyone to love the Journal; merely to perceive its positive achievements in spite of constraining circumstances.
A Tale of Two Professions — Civil Engineering and Computing

John M. Bennett*

Two of Professor Hawken's interests were the development and public image of engineering as a profession and the analysis of structures. The address is primarily concerned with aspects of these two interests, and their interaction with computing.

The role and history of numerical analysis is outlined, as is the development of computing methods used by structural engineers and the effects computers have had on these methods. Project control techniques in engineering construction and large software projects are surveyed. The address concludes with a discussion of the way computing can totally absorb a person's interest.

Keywords and phrases: History; computing and engineering profession; software trends; public opinion; engineering, science and computing; publishing patterns; numerical analysis; social effects of computing; civil engineering and computers; project control; software engineering; psychology of programming.


INTRODUCTION

The engineer is concerned with the handling and transformation of materials and energy, and the computer scientist with the handling and transformation of information. It is the purpose of this paper to draw parallels between the two professions, and their public images, to discuss the development of engineering computing techniques, and to show how the expertise of the engineering profession can provide many pointers to the way large computing projects can best be managed. Although the main emphasis is on structural engineering, much of the discussion applies to engineering in general.

THE COMPUTING INDUSTRY

In the last thirty years, computing has grown to be one of the world's largest industries. Some idea of its current size may be gauged from a comparison with the automobile industry. In the US, the value of computer sales is over a quarter of automobile (ex factory) sales of passenger cars. And the value of computer shipments from US manufacturers is increasing at about 15 percent per annum (Phister, 1976), while the value of automobile sales has been dropping.

Hardware sales figures alone give a misleading idea of the full impact of the industry. In Australia, for example, hardware sales represent about a quarter of the total expenditure of the computer industry if the salaries of everyone employed by vendors, suppliers of ancillary services and user computer installations are included. This total expenditure is about $1,500 million — about 1.5 percent of the Australian GNP (Thornton and Grace, 1980; Beardon, 1980; Bennett, 1980).

SOFTWARE TRENDS

During the fifties and sixties, most of the effort of computer professionals was devoted to extending the use of computers to new applications. However, the situation has changed, and systems design and the construction of systems programs now plays a much larger part in the activities of the professional.

Progress in electronics technology has been the primary driving force in the growth of the industry, and has resulted in an annual increase of computing power per dollar of about 60 percent (Dolotta, 1976). With this reducing cost of computing power has come an improvement in reliability (mean time between failures) of about 20 percent per annum — and with improved reliability has come the ability to undertake much larger projects with confidence. However, over the years, programmer productivity has increased at a much lower rate — at only 3 percent per annum, according to one writer (Dolotta, 1976) — despite the much vaunted advent of high level languages such as Fortran and Cobol.

The use of these high level languages has paid off in portability, as to a large extent, they have made programming machine independent. Moreover, as programmers can be expected to make one mistake for every few hundred instructions or program statements written down, and one high level language statement typically compiles into three to five machine language instructions (Brooks, 1975), the number of simple mistakes made in writing a program in a high level language should be fewer than would have been made if the same program had been written in assembly language. (These remarks apply to locating and correcting simple mistakes. Locating logical blunders is much more time consuming.)

Perhaps the greatest central problem in the computer field is the inability to estimate accurately how much effort

*The author is with Basser Department of Computer Science, University of Sydney, NSW, 2006. Manuscript received 25 March 1981, and revised 28 May 1981.

a programming project will take, and to ensure that the project will be completed by an agreed date.

There are many parallels between large software projects and large engineering projects, and so it is instructive for computer specialists to examine the history of large engineering project control — a theme which will be treated at greater length later in this paper. It is important to note here that the parlous state of the art of controlling large software projects does little to help the computer specialist in his quest for professional recognition.

It was this realisation which led in the late sixties to the introduction of the term “software engineering”. The term was originally introduced to underline for builders of software systems the need to adopt the attitude of the professional engineer towards project control, and to achieve a similar record for successful delivery of large systems on time and to specification.

CURRENT PUBLIC OPINION OF THE TWO CULTURES

The rift between the two cultures — the humanities on the one hand, and science and technology on the other — is a very real one. Engineering and computer specialists, being technologists, are both held by the champions of the humanities to be “affected by the myopia of numeracy” (Hill, 1980). In rebuttal, the eminent statistician M.G. Kendall has referred with scorn to “the veneer of protective mimicry bestowed by an education in the humanities”. The battle continues...

The public’s view of the engineering profession in the UK of a quarter of a century ago is apparent from B.V. Bowden’s plea in 1936 that it is necessary to “get rid of the tradition that makes the classics ‘U’ but technology ‘non-U’ and which values knowledge only if it is of no practical use” (Bowden, 1936). More recently, the Finniston Report (Engineering our Future, 1980) quoted the widespread popular view, quantified by a national opinion poll, that “an ‘engineer’ was someone doing manual work, probably with machinery”. According to the opinion poll, only 13 percent of respondents associated the title with design or research work at a professional level.

The Finniston Report points out that most engineers fulfil the characteristics of professional occupations in that they are required to be expert in a particular area for which practice requires a high level of theoretical foundation; their body of knowledge is clearly definable; and they are accountable for their decisions. However, unlike other professions such as medicine and law, engineers are mostly employed by large organisations, and they do not come into direct contact with the public, by whom the nature of their activities is not well understood.

A similar remark applies to computer specialists, who have the added disadvantage of being comparative newcomers to the professional scene. Computer specialists are even still in search of a suitable distinguishing title. The title computer scientist, for example, is misleading, as it implies that computing is a science — i.e., concerned more with analysis than design. (Concern with design [Simon, 1969] is the principal mark which distinguishes the professions from the sciences.) Perhaps the European title “informatics” would be the most suitable: however, its adoption by the English speaking world seems unlikely.

Statutory recognition in the form of a requirement that some documents can be certified only by a professionally qualified individual are perhaps more important in the eyes of the public than any other trappings of professionalism. At present, neither the computing profession nor the engineering profession has achieved this form of recognition in Australia. In the case of computing, no doubt if privacy laws call for the licensing and inspection of data banks, a requirement for certification by suitably qualified computer specialists will result in an enhanced professional image. However, the inevitable concomitant of this form of recognition is registration with state boards, a requirement which may prove to be a two-edged sword.

TRADITIONAL POPULAR IMAGES OF ENGINEERING

In ancient times, the nearest equivalent to the engineer was the smith, the worker with metals. Stories which have come down to us describe many of the mythological smiths as being physically deformed. Thus the biblical Jacob, who was connected with the Kenites, travelling tinkers of the mineral rich Wadi Arabah (roughly, the Jordan valley), was lame. So was Hephaestus (his equivalent Roman deity was Vulcan), who has also been depicted as a hunchback on Greek vases. Hephaestus was the constructor of various “ingenious works and automatical figures” including “two golden statues which not only seemed animated but which walked by his side and even assisted him in the working of metals”.

The attitude of Plato to engineering skills was a curious one (Lindsay, 1974) and probably has had a much greater influence on modern day thinking than most people realise. This attitude is best illustrated by Plutarch’s account (Clough, 1864) of the contribution of Archimedes to the defence of Syracuse against the Roman general Marcellus, almost a century and a half after Plato’s death.

In response to a request from the Syracusan king, Hiero, to show how some of his geometrical ideas could be put to practice, Archimedes had built various contrivances for the defence of Syracuse. Although Archimedes’ devices did not save the city, they greatly prolonged the siege. However, he had been reluctant to do so because Plato had expressed so forcefully his opposition to the use of mechanical devices as an aid to pure intelligence. It was because of Plato’s attitude that, as Plutarch put it, “machines became separated from geometry and, repudiated and neglected by philosophers, took its place as a military art”. No doubt, the reluctance to set up technological faculties shown by universities until quite recently owes much to the persistence of this attitude.

The skills and technical arts of the Romans made possible their bridges, roads, fortifications, aquaducts and imposing public buildings. In France and the German states, it was the monks who preserved these skills, as the monasteries and churches of the Middle Ages bear witness. They were in due course passed on to the builders of the great cathedrals and town halls of Europe, and particularly of Northern France (Cubberley, 1948). It was about the middle of the eighteenth century that secular engineering schools began to appear. In France, the Ecole des Ponts et Chaussees was founded in 1747 and in the German states a trade school was founded in Brunswick in 1745. The United States followed a little later — the West Point Military Academy was founded in 1802. However, universities everywhere resisted the introduction of technology.

THE ENGINEER AND THE SCIENTIST

To come to modern times, perhaps one of the most significant remarks concerning society’s view of the
TRADITIONAL POPULAR IMAGES OF COMPUTING

The computer field is currently very much under the microscope, primarily because of community concern with unemployment (Bennett, 1980) and privacy (Law Reform Commission, 1980). So any comment on the popular view of the image of the computing profession should take into account the current debate on the social implications of computers.

The popular image of computers and computer-like devices is better appreciated if we examine those age-old dreams of mankind which today because of computers, are closer to realisation than ever before. At least three separate motivations for the construction of computers can be discerned — the desire to simulate human and animal performance, the search for statues and heads with the gift of prophecy, and attempts to predict the motion of the heavens (Cohen, 1963).

An essential aspect of the move towards the realisation of these dreams has been the increasingly numerate basis for decision-making — a trend not universally approved. Edmund Burke (1729-1797) voiced what was probably a common view of his time when he said: "The age of chivalry is gone. That of sophists, economists and calculators has succeeded: and the glory of Europe is extinguished forever".

A popular manifestation of the reaction to automation in its early form was the Luddism of the period 1811-1816 (Thomis, 1970). The representation of this movement as being a reaction against automation does not bear close examination. At the time of the Luddites, there was an economic recession resulting from changing trade conditions (there had been an interruption of trade with the US) and assaults on machines appear to have been used as a weapon against employers who were using the recession as an excuse to depress wages. Thus breaking an employer's machines was a form of collective wage bargaining, trade unions being then illegal.

However, the popular view of the cause of the Luddite incidents, even though it may not be correct, plays a major part in the widespread reaction to automation today — both in the office and in the factory.

CIVIL ENGINEERING AND COMPUTING

Until the eighteenth century, the concept of engineering design as we know it seems to have played little part in the construction of large structures. The main factor determining the dimensions of a new structure appears to have been precedent — if a similar structure has stood the test of time, then it should be used as a model. Thus the Roman architect Vitruvius in detailing a suitable education for an architect, considered that skill at mathematics went beyond what was required, though perhaps it was desirable as one of the branches of knowledge with which every educated man should have had some contact.

The gradual emergence of the use of mathematical techniques for structural design makes an interesting story. Much of the available theory of elasticity (Todhunter and Pearson, 1960; Timoshenko, 1935) remained unused by civil engineers for a long time after it was published. There were perhaps three reasons for this, viz:

i) The non-availability of dependable materials. The advent of wrought iron and mild steel (the Bessemer process was first used in 1856 and the open hearth furnace in 1865) changed this picture. These materials not only had predictable properties, they also had the advantage that, because they could be rivetted, they could be readily used for the fabrication of large structural components.

Until the eighteenth century, the concept of engineering design as we know it seems to have played little part in the construction of large structures. The main factor determining the dimensions of a new structure appears to have been precedent — if a similar structure has stood the test of time, then it should be used as a model. Thus the Roman architect Vitruvius in detailing a suitable education for an architect, considered that skill at mathematics went beyond what was required, though perhaps it was desirable as one of the branches of knowledge with which every educated man should have had some contact.

The gradual emergence of the use of mathematical techniques for structural design makes an interesting story. Much of the available theory of elasticity (Todhunter and Pearson, 1960; Timoshenko, 1935) remained unused by civil engineers for a long time after it was published. There were perhaps three reasons for this, viz:

i) The non-availability of dependable materials. The advent of wrought iron and mild steel (the Bessemer process was first used in 1856 and the open hearth furnace in 1865) changed this picture. These materials not only had predictable properties, they also had the advantage that, because they could be rivetted, they could be readily used for the fabrication of large structural components.

ii) The lack of suitable design tools.

iii) The resistance of the existing design tradition.

In fact, pure science and technology flourish best in close collaboration. For example, the investigations into latent heat carried out by Black, a Glasgow professor of natural philosophy, led his friend James Watt to the condensing steam engine. The subsequent development of the reciprocating engine, which powered the industrial revolution, was carried out almost entirely by self-educated men.

"For many years the science of thermodynamics owed more to the steam engine than the steam engine owed to thermodynamics" (Bowden, 1956). Similar statements can be made of many other technical developments.

TRADITIONAL POPULAR IMAGES OF COMPUTING

The computer field is currently very much under the microscope, primarily because of community concern with
ii) The absence of certain key pieces of theory. For example, Telford's Menai bridge owed more to tests than theory (Paxton, 1980). Though the problem of the undulation of the deck generated many theoretical investigations, the aerodynamic flutter which caused this undulation was not properly understood until the Tacoma Narrows bridge failure 130 years later (Straub, 1964).

iii) Computational difficulties. The labour of computation inevitably discouraged the use of any but statically determinate structures. As it is statically determinate, the truss rapidly superseded the tubular bridge of which the Britannia bridge was the paradigm. Truss bridges, developed in the 1840s, mainly in the US and Russia, were also lighter and were more familiar to engineers used to timber construction.

The French Genie officers of the 18th century, mostly associated with the Ecole des Ponts and Chaussees, laid the foundation for most of the theory of elasticity as we know it today. Among the giants associated with the Ecole, many of whom are also known as mathematicians and physicists, were Coulomb, Navier, Cauchy, Saint Venant and Monge. Monge (1746-1818), a friend and confidant of Napoleon, deserves special mention (Bell, 1953). In addition to his contributions to structural theory, he is credited with having laid the groundwork of descriptive geometry—the engineering drawing of today.

Monge's techniques, which were regarded as a military secret by the French until 1794. They were used in 1799 by Marc Brunel, Isambard Kingdom Brunel's father, to specify the design of a new type of blockmaking machine for the Admiralty.

Much of the credit for bringing the design methods of French engineers to Britain must be given to Rankine, whose book "Applied Mechanics" appeared in 1858. Rankine will be remembered by today's engineers for his work on earth pressures; in fact, his interests and writings covered a much wider field (Hamilton, 1958).

The effect of the dependable properties of wrought iron and steel has been mentioned. The use of these structural materials quickly extended to ships and buildings. In the case of buildings, presentday engineers observe while using the new techniques led to a parting of the ways for architects and engineers, the former specialising in questions of appearance and the latter in structural design.

As trusses came into general use, the call for streamlined methods of analysis led to the introduction of graphic statics first described by the Swiss Kulmann in 1864.

Graphic statics was said to be superior to other computational methods because it was a visual method, which was held to give it advantages of the visual over the arithmetic approach of other computational methods.

Others who contributed to structural engineering computational methods during this period include Clapeyron (who first applied the principle of minimum energy to elastic structures and proposed the three moments theorem for the analysis of continuous beams); Maxwell (who first proposed the principle on which influence lines are based); and Menabrea and Castiglano (who analysed statically indeterminate structures by the principle of minimum energy).

Menabrea, a military engineer, became Prime Minister of Italy in 1869. His record of a lecture given by Babbage to the military academy at Turin in 1840 is the fullest description we have of Babbage's analytical engine. It was translated by Bryon's daughter, Ada Augusta Lovelace, who added the extensive translator's notes containing sample programs for the machine.

The principle of minimum energy was a very old one, and in the eighteenth century appeared to be an expression of a theological principle that God had created "the best of all possible worlds", which permitted the greatest effect for the least effort (Straub, 1964). Unfortunately, when applied to indeterminate structures, it called for a considerable amount of effort to solve the resulting linear simultaneous equations. With the benefit of hindsight, we can now see that, in cases where they would converge quickly, iterative methods of solution were clearly appropriate. So it is not surprising that, when Hardy Cross propounded his method of moment distribution (Cross, 1932), it quickly became the standard pre-computer technique for a wide range of indeterminate structures.

Southwell's relaxation technique, more general in character but demanding more judgement for its effective application, followed (Southwell, 1940). The relaxation technique has been extended to a wide range of physical problems in addition to the determination of stresses in engineering structures (e.g., Allen, 1954).

In 1944, Gabriel Kron (1944 a; 1944 b), then a General Electric employee, introduced the now commonly accepted network approach to adapt a network analyser (an analogue machine used by power engineers for computing current flows in networks) to solve structural engineering problems (Bennett, 1952; Alger, 1969; Happ, 1973). It provided the basis for Argyris's work, which was first published in 1954 (Argyris, 1960).

When Kron began to work with small digital computers in 1953, he proposed a method for dividing a problem into sections (the method of "diakoptics" or tearing of networks) to increase the size of computations which could be handled with the limited computer storage available to him. This method, when applied to continuous elastic media and stripped of Kron's mystique (his writing became increasingly obscure in his later years), was in effect the method of finite elements.

Argyris in 1954, and Martin and Clough in 1956 (Strang and Fix, 1960) are usually credited with being the first to introduce a working version of the finite element method for analysing elastic structures. It has now been extended to other areas, and its mathematical legitimacy, initially in doubt, seems to have been firmly established by identifying it as a special case of a long established mathematical technique, the Rayleigh-Ritz method.

As computers provide the means of solving directly large sets of simultaneous equations, it is not surprising that they are now widely used for virtually all types of structural calculations, including elasto-plastic calculations and structural optimisation (Spillers, 1980). Also, because of the complex nature of design specifications (legislated building codes, model building codes, consensus standards, proprietary and trade specifications, owner's specifications, etc. all have to be taken into account), the possibility of applying to structural engineering design problems techniques similar to those being used in the artificial intelligence field is currently being investigated (Fenves and Norabhoompipat, 1978).

Computer graphics techniques are now well established, as any recent graphics conference proceedings will testify (e.g., SIGGRAPH '80 Conference Proceedings [1980]).
And as usual drafting costs mount, so computer graphics packages become more attractive economically. Moreover, the extension of these packages to computer assisted manufacturing (i.e., the control of machine tools, etc.) is now accepted practice in mechanical engineering (Elliot, 1978).

Many available packages are interactive — i.e., they permit a designer to alter his designs directly on the face of a CRT display, and to have the results of the alteration displayed. This combined visual and arithmetic approach is also used successfully to check for errors in data prepared for input to large structural programs, and to assist in the layout of mechanical services in buildings.

A 1974 publication (Pilkey et al., 1974) lists about 600 structural mechanics computer programs as being available for general use, and describes current US software dissemination practices (Schiffman, 1974). These programs cover a wide range of applications and are easily obtainable. So, before a new programming project is started, some effort should be expended to check whether a suitable software package is available commercially. Unfortunately, a recent examination of the Australian scene (Bennett, 1978) shows that commercially available technical software packages are little used in this country, and probably there is a considerable duplication of effort as a result.

PROJECT CONTROL

The story of the Tower of Babel (Genesis 11: 1-8) is the tale of a major project using the newest technical innovations of the day (fired bricks for stone, bitumen for mortar), an apparently infinite supply of labour and a clearly defined objective. What went wrong? The Lord deemed the project to be overambitious, and saw to it that labour had to be drawn from many sources with no common language. So co-ordination was impossible.

The limit to project size which the Tower of Babel story illustrates is still with us today. It is still set by the difficulties entailed in ensuring that all members of the project team "speak the same language", and can only be raised by the use of accepted procedures and standards understood by all team members.

In recent times, Telford (1757-1834) is given credit for a major improvement in project organisation in connection with his canal building activities (Pollard, 1965). He worked with a small group of contractors who, after a time, became to know what was expected of them. He introduced a system of monthly payments with retention of definite sums as a guarantee of satisfactory workmanship and punctual completion, and required a period of maintenance during which a contractor was responsible for the state of the new work. As far as possible, managerial and administrative responsibilities were vested in the contractors, who, in turn, employed subcontractors where necessary, supplying them with any necessary plant (cranes, etc.). The engineer responsible for the project was thus freed to devote most of his time to technical problems.

What Telford did for canals, Joseph Locke (1802-1860) did for railway construction (Devey, 1862). From the point of view of the investors, this form of project control brought with it the advantage of cost predictability. Much of Locke's reputation rested on his never being more than a few percent out in his estimates.

As a result of these techniques, it was possible to bring much greater total effort to bear on a project at any one time than had previously been the case. However, coordination difficulties still remain a problem, and have led to the introduction of two techniques in the 1950s— the critical path method (CPM) and the programme evaluation and review technique (PERT). An undertaking that PERT will be used (with the help of a computer, of course) is now a contractual requirement for many US government projects.

CPM, introduced originally for the control of chemical plant maintenance operations, provides a method of determining quickly whether delay in any particular operation will delay the completion date of the project as a whole, so that extra effort can be mustered for these time-critical operations if necessary. PERT, introduced to assist in the control of the US space programme, permitted the estimates of the individual operation times to be given as upper and lower bounds and provided a statistical picture of the distribution of overall delays if the incidence of individual delays was independent. (When applied to civil engineering projects, the assumption of independence of delays in individual operations often cannot be sustained. For example, when it rains, all outdoor operations are delayed.)

CONTROL OF COMPUTER SOFTWARE PROJECTS

Only in the last decade has the nature of project control problems in the computer software field been the subject of widespread discussion. Various proposals have been made concerning the most effective way to organise programming work and programming teams (e.g., Weinberg, 1971; Brooks, 1975), most of these being concerned with comparatively small projects. They include the following:

i) The use of small teams of programmers of proven ability.

ii) Egoless programming. This approach calls for the checking of a program by someone other than the author before machine testing. The name derives from the belief that this process will discourage individual programmers from becoming personally identified with their creations.

iii) Providing proper support for the individual experienced programmer. The experienced programmer becomes the team leader, the team being limited to a size which he can supervise personally.

iv) Using only programs of proven correctness. The term "correctness" is used in the mathematical sense: if this restriction were accepted literally, the computer industry would not exist.

v) Restricting individual program modules to a size which can be completed and tested in a fortnight. This restriction ensures that supervisors know early about overruns.

vi) Structured programming. This technique requires the use of simple structures to interlink program modules, themselves composed of simple structures.

vii) Top-down programming. This approach consists of first writing a control program, then adding the various modules which can thus be tested in the final environment in which they will operate.

viii) Bottom-up programming. This approach consists of constructing a large program from existing modules which, where possible, have been tested in other contexts.

All these proposals relate to the construction of programs, and assume that what is to be done is clearly specified. Unfortunately, this is frequently not the case. Often, particularly if the top down approach is adopted, awkward...
sections of a project are left for later definition so as to
“get on with the job”. When these sections eventually have
to be defined, they may often necessitate the scrapping of
much of the work already done.

There is a growing recognition (Brooks, 1975) that,
for a large project, there is no substitute for conventional
engineering practice. Thus the sequence of operations con­
sisting of specification, tendering, constructing to specifica­
tion, and acceptance testing by an independent group or the
controlling authority, followed by a period of warranty
(during which the contractor accepts maintenance respon­
sibility), the traditional engineering approach, is the most
satisfactory one. Departures from specification should be
separately costed and made the subject of a formal contract
variation. Inevitably, as often happens with civil engineering
projects, unforeseen difficulties will arise, and contracts
should be written accordingly.

CONCLUDING REMARKS

I have endeavoured in this paper to develop two
themes — the status of the profession of engineering and its
interaction with available computing tools. I have outlined
parallel developments in the computing profession which,
though in its present form much younger than engineering,
have origins which are as ancient. And I have tried to show
that we in the computing field have much to learn from
the hard-won experience of the engineering profes­
sion.

Engineering has been defined (Wellington, 1888) as
“the art of doing well with a dollar what any bungler can
do with two, after a fashion”. It would be appropriate for
the computing profession to adopt a similar definition as
a hallmark of the professional acceptability of our work.

REFERENCES

Albany, N.Y.: Mohawk Development Service.
ARGYRIS, H.J., and KELSEY, S. (1960), Energy Theorems and
BEARDON, C. (1980), The Political Economy of Computing in
Australia. Journal of Australian Political Economy, No.
7, April.
BELIET, E.T. (1953), Men of Mathematics. Harmondsworth:
Penguin.
BENNETT, J.M. (1952), Some Engineering Applications of Digital
Computing, PhD dissertation, Univ. of Cambridge.
BENNETT, J.M. (1978), Scientific and Technical Computing in
Australia: Position paper prepared for the Australian Science
Department of Computer Science, University of Sydney.
BENNETT, J.M. (1980), Computers and Associated Technological
Trends — Their Effects on Employment Patterns. Australian
BOWDEN, B.V. (1956), Proposals for the Development of the Man­
chester College of Science and Technology, Manchester: Jesse
Broad.
COHEN, J. (1963), Automata and Myth in Science, History Today,
Vol. 13, No. 5.
ada.
COLAABAVA LA, F.D. (1976), Tantra: the Erotic Cult., New
Delhi: Orient Paperbaks.
CROSS, H. and MORGAN, N.D. (1932), Continuous Frames of Re­
inforced Concrete. New York: Wiley.
CUBBERLEY, E.P. (1948), The History of Education, London:
Constable.

ELLIOTT, W.S. (1978), Interactive Graphical CAD in Mechanical
Engineering Design, Computer Aided Design, Vol. 10, No. 2,
March.

Engineering our Future: Report of the Committee of Inquiry into

FEIGENBAUM, E.A. (1977), The Art of Artificial Intelligence 1:
Theme and Case Studies of Knowledge Engineering, Report
STAN-CS-77-621, Stanford University, Stanford, Ca.
FENVES, S. and NORABHOM Pipes, T. (1978), Potentials for
Artificial Intelligence Applications in Structural Engineering
Design and Detailing. In Proc. IFIP TC-5 Working Confer­
ence, Grenoble, Amsterdam: N. Holland.
HAMILTON, S.B. (1958), Building Materials and Technology. In A
History of Technology, Vol. 5, edited by E. Singer et al. Lon­
don: OUP.

HAMP, H.H. (ed.) (1973), Gabriel Kron and Systems Theory, Schen­
cedyt: General Electric.

Profession of the Civil Engineer, edited by D. Campbell-Allen
and E.H. Davis. Sydney: Sydney UP.

HAWKEN, R.W.H. (1941), Economy of Purchase with Tables. Bris­
bane, University of Qld, Faculty of Engineering.

HAWKEN, R.W.H. (1926), The Location of City Bridges, Presiden­
23 October, p. 245.
KRON, G.J. (1944 a), Equivalent Circuits of the Elastic Field. J.
Franklin Inst., Vol. 234, No. 6, December.
KRON, G.J. and CARTER, G.K. (1944 b), Network Analyzer Solu­
tion of the Equivalent Circuits for Elastic Structures. J.
Franklin Inst., Vol. 238, No. 6, December.


MESAROVIC, M. and PESTEL, E. (1975), Mankind at the Turning
Point. London: Hutchinson.

In Thomas Telford: Engineer, edited by A. Penfold. London:
Thomas Telford.

PHISTER, M. Jr. (1976), Data Processing Technology and Econo­
mics, 1975-1978. Supplement. Santa Monica: Santa Monica
Publishing.

PILKEY, W. et al. (1974), Structural Mechanics Computer Pro­
grams. Charlottesville, Va.: University of Virginia.

POLLARD, S. (1965), The Genesis of Modern Management. Lon­
don: Arnold.

PRICE, D.J. de Solla (1975), Gears from the Greeks. New York:
Science History Publications.

PRICE, D.J. de Solla (1962), Science since Babylon, New Haven:
Yale UP.

SCHELLMAN, R.L. (1974), Software Dissemination Practices and
Organisations. In Pilkey, op. cit.

SIMON, H.A. (1968), The Sciences of the Artificial, Cambridge,
Mass.: MIT Press.

SMILES, S. (1905), Industrial Biography: Ironworkers and Tool-

Harvard UP. (Godkin Lecture 1960).

SOUTHWELL, R.V. (1940), Structural Mechanics Computer Pro­
grams. Charlottesville, Va.: University of Virginia.

STRANG, G. and FIX, G.J. (1960), An Analysis of the Finite Ele­

STRAUB, H. (1964), A History of Civil Engineering, Cambridge,
Mass.: MIT Press.

THOMIS, M.I. (1970), The Luddites, Newton Abbot: David and
Charles.

THORNTON, B.S. and GRACE, J.C. (1980), Section of
Engineering our Future: Report of the Committee of Inquiry into
A Tale of Two Professions


**BIOGRAPHICAL NOTE**

The author received his B.Eng. degree from the University of Queensland in 1942. After war service with the RAAF Signal Corps he completed a Ph.D. at University of Cambridge in 1949, and afterwards was a Computer Specialist with Ferranti Ltd, Manchester, England during 1950-1955, before joining Sydney University as Senior Numerical Analyst, which position he held until 1961. He then became Professor of Computer Science at the University, a position he has held since then.

Professor Bennett oversaw the construction of the SILLIAC computer at Sydney University, and headed the then Basser Computing Laboratory, later to become the Basser Department of Computer Science. He was the Founding President of the Australian Computer Society in 1966-67, and has been a Vice-President of IFIP since 1975. He has played an active advisory role on computing policies and their social effects.
On the Design of Process Assigner for Distributed Computing Systems

R.K. Arora,* S.P. Rana,* and N.K. Sharma*

In distributed operating systems, an important facility to be provided is the process assigner. The assigner maps processes of a distributed program to processors for execution so as to optimize system performance. In this paper we have stated the properties of assigner and formalized its conceptual structure. Three implementation strategies are then presented for the assigner, and their merits and demerits discussed.

Key words: Distributed operating systems, process assigner, distributed program, load sharing, centralized versus distributed control.

CR Categories: 4.30, 4.32

1. INTRODUCTION

In recent years, we see a trend towards the replacement of conventional computer systems based on a large central processor by distributed computing systems. The main driving force behind the evolution of distributed systems is that they can offer high degree of availability, extensibility, reliability and concurrency. These systems present computing environment different to single processor or multiprocessor systems. As such, new methodologies and techniques are required for the design of distributed operating systems and for expressing applications as distributed programs.

The essential characteristic of a distributed computing system is the existence of a global operating system which co-ordinates the entire activities in the system. The global operating system presents the distributed system to users as a single virtual machine capable of executing distributed concurrent applications (Enslow, 1978; Kemen and Nagel, 1978).

Recently much research effort has been focused on the design of distributed operating system; but the attention has been mainly focused on issues like interprocess communication, synchronization, creation of processes, etc. In order to exploit the naturalism of applications (distributed structure) and the benefits of distributed computing, the applications must be expressed into distributable components (called processes) assignable to different processors before execution. Since one of the design objectives in distributed operating systems is transparency, the process assignment function must be performed by the system. The agency in the distributed system that determines process assignments is hereafter called the assigner.

In the present paper we identify the issues involved in designing an assigner. A conceptual structure and the properties of assigners are described in section 2. There are three implementation strategies: namely, (i) centralized assigner, (ii) multiple copies of assigner and (iii) distributed pipelined assigner, are presented in section 3 and their relative merits and demerits are examined.

2. STRUCTURE AND PROPERTIES OF ASSIGENER

This section is devoted to the description of the logical structure of assigner and the specification of basic properties that are to be satisfied by the assigner regardless of the implementation strategy adopted.

We begin with a brief description of the model of distributed computing in order to facilitate further discussion. The model that follows has earlier been adopted by several authors in different contexts — to mention a few (Kemen and Nagel, 1978; Nelson and Gordon, 1978).

The Distributed Computing System

A distributed computing system is a collection of

```
CSN: Communication subnetwork P1 P2 P3 P4: Processors, M1 M2 M3 M4: Memory modules.
```

Figure 1. Pictorial depiction of a distributed system.

*The authors are with the Computer Centre, Indian Institute of Technology, Hauz Khas, New Delhi 110016, India. Manuscript received 26 May 1980, and revised 21 November 1980.

The Structure of Assigner

A Distributed Program

A distributed program is expressed in terms of interacting components where each component corresponds to some process in the system. Users can directly write programs if a distributed programming language is made available. Attempts in this direction are seen in (Brinch-Hansen, 1978; Feldman, 1977 and Liskov, 1979).

In order to execute a distributed program, the corresponding processes are to be associated to unique processors where they are created and executed subsequently.

Assignment of Processes

The dynamic assignment of processes to processors is a pertinent issue — and the least understood — in the design of distributed computing system. To distribute applications in a satisfactory manner, techniques are required for choosing best process assignments among several choices. Further, there is a need for evolving mechanisms to perform the assignment function. Two options are there: assignment function may be performed either by user or by system itself (Jensen and Bocbert, 1976).

Process Assignment by Users

The McRoss system (Thomas et al., 1972) is an earliest example of systems that allow the dynamic assignment of processes. In McRoss however, the reassignment is performed through user interaction only. Several other works have been reported that advocate that the process assignment must be done by users themselves. An example is $\mu$-Machine (Thalman, 1978), which allows the users to declare process assignments in a configuration section.

Process Assignment by System

One of the important characteristics of distributed systems — as pointed out in (Enslow, 1978) — is to have system transparency, that is, users must know the type of services available in the system by names and not by their locations. To achieve transparency, it is necessary that the process assignment is performed by some agency (to be called the assigner, here) of the distributed operating system. There has been a little progress towards the design of such an agency.

Work has been reported on techniques to compute optimal assignments (Stone, 1977; Arora and Rana, 1978, 1979). However, none of the above works touch the issue of designing an assigner. We intend to fill this gap by discussing the structure, properties and implementation strategies of the assigner.

The Structure of Assigner

Conceptually, the assigner can be envisaged as constituting three portions: the front-end portion, the computational portion and the back-end portion. The front-end portion gets information, requisite for assigner function, from the relevant agency — and thus, provides an interface for accessing the assigner. The computational portion computes process assignments by employing suitable algorithms. These algorithms may generate optimal or near optimal assignments, depending upon the model selected. The back-end portion communicates the computed assignment to agencies responsible for controlling execution and updating load status.

The assigner expressed in the form of a process is formally outlined below in concurrent Pascal-like notation. The data structures, communication primitives and the processes interacting with the assigner are elaborated in the Appendix.

```
BEGIN
  ASSIGNER
  VAR
    X : input-to-assigner;
    Y : assignment;
    L1, L2 : load-vector;
  BEGIN
    CYCLE
      BEGIN
        RECEIVE X FROM configurator;
        RECEIVE L1 FROM load-recorder;
        Compute Process assignment;
        SEND Y TO execution-controller;
        SEND L2 TO load-recorder;
      END
    END
  END
END
```

THE PROPERTIES OF ASSIGNER

Regardless of which strategy is followed for implementation, the assigner must always meet certain specifications and it must satisfy some other desirable characteristics to a reasonable degree. These are listed below.

Obligatory properties

(a) All distributed programs logged into the system must get access to the assigner in a finite interval of time.
(b) The assignment function for a particular program must be performed in finite time interval.
(c) All processes of a distributed program must have been assigned to processors after the completion of assignment function.
(d) A process must be assigned to one and only one processor and the process must be executable by that processor.

Desirable properties

(a) The underlying algorithm must generate assignments that optimize a chosen performance criterion.
(b) The overhead of implementing the assigner must be
On the Design of Process Assigner for Distributed Computing Systems

The load-recorder is implemented at the location of the assigner and load information is maintained centrally. The centralized strategy renders it possible to distribute load in an optimal fashion.

The Network Computer TECHNEC described in (Huen et al., 1977) employs the centralized strategy. The object code of a sequential program after compilation in a pipelined manner is sent to a unique processor which partitions the code into clusters and assigns clusters to processors. The TECHNEC system represents a specific configuration where users interact through only one processor, called system processor. The centralized approach becomes a natural choice for such systems.

In general, users interact with a distributed system simultaneously from different nodes. The centralized approach in such a system is not desirable for at least the following reasons:
(a) Being a single assigner in the system and all programs requiring access to it prior to execution, the assigner becomes a highly shared resource and thus, a bottleneck in the system. The contention for assigner degrades systems performance considerably.
(b) The reliability of the assigner is limited by the reliability of the processor holding the assigner.
(c) All processors send messages to the assigner for performing assignment function. This causes a heavy traffic activity with the host processor of assigner, again degrading system performance.

Distributed Control Strategies

As emphasized by Enslow (Enslow, 1978), the performance of the system must not be seriously degraded by the failure or overload of a resource that has a permanently assigned system control function. To ensure the above characteristic, the system must have multiple loci of control. We now explore the strategies permitting distributed control for implementation of the assigner function.

The moment a distributed control is envisaged, we are faced with issues not present in centralized control. According to G.Le Lann (Lann, 1977) problems to be solved with distributed control are:
— control must be achieved without knowledge of the global state. Therefore, what is needed is that each entity behaves according to some algorithm working on an approximation of this state. In spite of this uncertainty, these algorithms must be such that entities are kept in a legitimate state and that the global system behaviour is a convergent process;
— there is no entity which is, a priori, in charge of performing the control.

The above comments imply, in the case of an assigner, that the assignment must be performed locally with local information to accomplish the global objectives of distribution of processes. Obviously, the assignments performed by local optimization strategies need not be globally optimal, but the effort must be to get the best assignments (according to some chosen criterion) possible.

The two strategies given below assume distributed control. In discussing these, we assume that the processors in the system are connected via a virtual ring (it may or may not be a physical ring) such that for each processor there exist an immediate predecessor and immediate successor processors.
On the Design of Process Assigner for Distributed Computing Systems

(ii) Multiple copies of Assigner

In this approach, there are replicated copies of the assigner resident in different nodes (Figure 2b). An assigner only caters to the requests generating from its host processor.

The structure of an assigner is the same as in the centralized approach. All copies of the assigner run independently and concurrently except that they share the global information of type load-vector via a load-recorder. The load-recorder is implemented in a distributed fashion, too. We refer to two schemes, viz (a) mutual listening, and (b) the circulating vector techniques, for providing load-status information to nodes in a distributed fashion (Lehon et al., 1976).

The multiple copies approach may be globally unstable due to the fact that more than one processor after finding a processor lightly loaded may simultaneously relegate their load to make it heavily loaded. The stability can, however, be achieved by imposing additional conditions. As shown in (Lehon et al., 1976), the stability conditions can be computed according to hardware performances and processing requirements and achieving stability might require a trade-off with the traffic activity incurred by circulating load information. In simple terms, the stability of assignment function relates directly to the frequency of load-information circulated to the processors. The less the time interval between successive load-vectors received, the more exact the processors have knowledge of the system load status.

The implementation of assigners, as such, has not been suggested in any previous work; however, the above strategy has been suggested quite often in similar contexts. In (Lehon et al., 1976), we find the above strategy employed for automatic load-sharing, but at job level.

In another system named ARACHNE (Finkel et al., 1980), the resource managers — a utility process of the distributed operated system proposed — have been implemented in the similar manner as above.

Few more points are observed regarding the above approach:

(a) No communication cost is involved in accessing the assigner.

(b) As pointed out earlier, the above strategy may be globally unstable. To guarantee stability would require additional overhead in terms of message traffic and this may offset the advantages of this approach.

(c) The approach presents a modular and reliable design. The failure of a processor affects the requests generated from it and the processes which are assigned to it — yet unexecuted — till the time when its failure is known to other processors. Once the other processors know about a failure, they proceed with the assignment function with the remaining processors.

(d) All processors have their independent assigners and thus, the assignment function at all processors proceed concurrently.

(iii) The Distributed Pipelined Assigner

In this strategy, there is a logically single assigner; however, it is physically realized in a distributed fashion. The assigner is constituted by several identical components — to be called local-assigners. Each processor in the system, holds a local-assigner (Figure 2c). The program input at a particular node access the assigner through its local-assigner. Subsequent to activation of a local-assigner by

Figure 2b. Multiple copies of assigner.

Figure 2c. A distributed pipelined assigner.

input from its host node (to be called originating node), the computation of the assignment of the program in question progresses in a pipelined fashion as explained below.

The local-assigner of the originating node computes assignment by employing a two-processor assignment algorithm (for details, see [Stone, 1977]), by taking the originating node as one processor and assuming the rest of the system as the other processor. Stone has shown that the above algorithm would associate to the former processor, the processes which are associated to it by a globally optimal strategy (viz., by considering the N-processors problem; N being the total number of processors in the system).

Thus, by the above algorithm, the local-assigner separates out the processes to be assigned to its host processors. Consequently, it leaves out a reduced problem consisting of the left-over processes.

The left-over processes are now considered for assignment on the remaining processors. To accomplish this, the local-assigner interacts with the local-assigner of the next processor on the ring and communicates to it the following information:

- the identity of the originating node, and
- the reduced problem (input-to-assigner).

The latter local-assigner then considers the problem submitted to it for assignment on its host processor and its successors up to the immediate predecessor of the originating node. The local-assigner proceeds in the same manner as above and interacts with the next local-assigner except in the cases where,

- either all processes of the given program have been assigned,
- or the current processor is immediate predecessor of the originating node — in which case all left-over processes would trivially be assigned to the current processor.

The load-vectors may be circulated as in approach (ii). The local-assigners do take into consideration the latest load information available on their host nodes. It should be noted here that all local assigners always have accurate knowledge of the load on their respective host nodes.

In the above implementation of the assigner, the local-assigners receive input from two sources — namely, the host processor and the local-assigner of the immediate predecessor node. The requests from the two sources can be entertained by some pre-determined discipline. We suggest that the requests accruing from a local-assigner should be given priority over the request from the host processor. (This suggestion stems from the intuitive feeling that the former requests would require less processing).

The distributed pipelined approach offers the following advantages:

(a) Since processes to a node are assigned only by its local-assigner, a local-assigner always has an accurate information about the load on its host node. The above fact renders it possible to achieve the local optimization of load distribution objectives. (The assignment is not globally optimal because firstly, the local-assigners do not have accurate load information regarding other processors and secondly, the underlying algorithm to compute assignment generates near optimal assignments. For details of the algorithm, we refer to [Stone, 1977].

(b) The unstable situation (pointed out in approach [ii]) where a processor is heavily loaded by other processors simultaneously reallocating load after finding the former processor lightly loaded, cannot occur in the present approach simply because no processor can assign load to other processor.

(c) The components of the assigner are autonomous and proceed independently. Thus the concurrency is achieved at least up to the same extent as in approach (ii).

(d) In the strategies (i) and (ii), after an assignment is determined, the messages need to be sent to all concerned nodes to inform them of the respective processes assigned. This communication overhead is eliminated in the present approach because of the processors computing their own assignment.

(e) Finally, an important advantage of the above strategy is its potential for surviving processor failures in the system. When a processor fails, the distributed operating system automatically creates another virtual ring excluding the failed processor.

The distributed pipelined approach seems to have one drawback. If there are significant delays involved between the successive services to a program by different local assigners, the response time of an assigner for a particular program could be quite high. On the other hand, we point out that the time taken by a local-assigner is bounded above by \(\frac{N-1}{N} \) th of time taken by the assigner in approaches (i) and (ii) for a particular program (N being the total number of processors). This is because a local-assigner executes a two-processor rather than a N-processor assignment algorithm.

4. CONCLUSION

Three implementation strategies for dynamic assignment of processes in a distributed computing system are presented. The merits and demerits of these have been discussed from various points of view. The centralized approach can be discarded for fully distributed systems that must provide distributed control of system functions. For better assessment of the two distributed control strategies presented in the paper their performances need to be evaluated by simulation or actual experiments. The authors are at present engaged in developing a simulated model of distributed systems. The model would be employed to investigate further the concepts discussed herein. The ideas set forth in this paper contribute towards the design of distributed operating systems in general and the design of the process assigner in particular.

ACKNOWLEDGEMENT

We are highly thankful to the referee and the Editor for their critical and useful comments that have helped us to improve upon the initial version of this paper. Part of the research work was sponsored by a grant from the Department of Science and Technology, India.

REFERENCES


On the Design of Process Assigner for Distributed Computing Systems


APPENDIX

The data structures, primitives and other processes used in the assigner process are elaborated below.

DATA STRUCTURES

(a) INPUT-TO-ASSIGNER

```plaintext
  type input-to-assigner
     = record
       pgn : program-global-name;
       pg : process-interaction-graph;
       rncost : array [1 .. N] of run-cost-array
      end

  type program-global-name
     = record
       pid : program-identifier;
       nid : node-identifier
      end

  type process-interaction-graph
     = record
       ps : process-set;
       edges : array [1 .. V] of comm-edge
      end

  type process-set = set of process-name

  type comm-edge = record
       pn1 : process-name;
       pn2 : process-name;
       comm-cost
      end

  type run-cost-array
     = array [1 .. N] of run-cost

  type assignment = record
       pgn : program-global-name;
       pnm : array [1 .. M] of process-node-map
      end

  type process-node-map = record
       pn : process-name;
       nid : node-identifier
      end

  (b) ASSIGNMENT
      var comm-cost, run-cost : real

      (b) SEND item TO process-name
      The SEND primitive enables the invoking process to transfer the data specified in 'item' to the process 'process-name'. The location of processes being transparent the transfer of data might be between local or remote processes.

      (b) RECEIVE item FROM process-name
      The RECEIVE primitive enables the invoking process to accept the data in 'item' from the process 'process-name'. Again, the communication could be local or remote.

(b) COMMUNICATION PRIMITIVES

(a) SEND item TO process-name

(b) RECEIVE item FROM process-name

(c) Vector LOAD-VECTOR

var comm-cost, run-cost : real

(b) ASSIGNMENT

(c) Vector LOAD-VECTOR

COMPUTATION

The computation involves assigning processes to nodes, scheduling the execution of processes, and managing the flow of data between processes.

(b) COMMUNICATION PRIMITIVES

(a) SEND item TO process-name

(b) RECEIVE item FROM process-name

(c) Vector LOAD-VECTOR

PROCESSES INTERACTING WITH ASSIGNER

(a) CONFIGURATOR

(b) LOAD-RECORDER

(c) EXECUTION-CONTROLLER

BIOPGRAPHICAL NOTE

R.K. Arora is a Professor of Computer Science and leader of Systems Software Group of ICL-2960 Computer System at Indian Institute of Technology, Delhi. Dr Arora was earlier associated with the Software Group at TIFR, Bombay in 1963-67. He has been a visiting research fellow at the Institute of Computer Science, University of London, during 1971-73. His current research interests include computer architecture, operating systems — design and evaluation, graph theory, arithmetic codes and distributed computing. Dr Arora is a member of the British Computer Society and Computer Society of India. Dr Arora received his M.A. in Mathematics from Delhi University in 1962 and Ph.D in Computer Science from IIT, Delhi in 1971.

S.P. Rana is a lecturer at Computer Centre, Indian Institute of Technology, Delhi and is currently doing research in distributed computing. His other research interests are scheduling theory and operating systems. He had his M.Sc in Operations Research from Delhi University in 1975 and Postgraduate Diploma in Computer Science from Kurukshetra University in 1977.

N.K. Sharma is currently working towards a Ph.D degree at Indian Institute of Technology, New Delhi. His research interests include operating system and languages for concurrent programs.
LOCAL GOVERNMENT COMPUTING IN THE GEELONG REGION

R. Philcox*

*The author is with the Division of Computing and Mathematics, Deakin University, Waurn Ponds, Victoria, 3217. Manuscript received 2 October 1980. Revised 22 December 1980.

A number of local government authorities in the Geelong area have increased their use of computers over the last decade. Although there is agreement that several of their application systems perform very similar tasks, any arrangements for pooling resources can be considered a problem which has yet to be solved.

Keywords and phrases: Local government computing, politics, technological decisions.
CR categories: 2.19, 2.44, 3.53.

INTRODUCTION

The Geelong Regional Commission was formally established by an Act on 24 May 1979. Its brief is somewhat vague covering as it does matters such as:

a) The participation of the people of the Geelong region in planning the future of the region and the nature and limits of such development.

b) The planning co-ordination integration and management of and for:

i) Development within the Geelong region.

ii) The improvement and enhancement of the nature and limits of such development.

iii) The provision of services and facilities and improvement of industrial, commercial and other undertakings..." (Acts, 1977).

The region "covers" an area of 2528 square kilometres ranging from Little River in the East, Elaine-Meredith to the North and to Lorne in the West. Within this area a number of local governments exist: the Shires of Bannockburn, Bellarine, Barabool and Corio; the Borough of Queenscliff; and the Cities of Geelong, Geelong West, Newtown and South Barwon. There are also numerous other authorities both large and small, the largest being the Geelong Water and Sewerage Trust and the Geelong Harbour Trust.

In 1979 the GRC decided to undertake "a preliminary evaluation of the possibilities inherent in the concept of co-operative computer usage by local authorities in the region" (McCalden, 1979), and, accordingly, a study entitled "Computer Use in the Geelong Area by Certain Statutory Authorities" was commissioned and carried out. The methodology used for the study was that of interview using a questionnaire as a means of structuring it.

The following list of questions formed the basis of the interview.

1. What data processing, including manual processing, does your authority undertake?
2. Do you use automatic data processing? If so, for what end in what forms?
3. If you have a computing machine, what are its characteristics? How many data processing staff are employed? What software is available and where was it developed?
4. If you plan to use a machine in the future, say in five years' time, have you drawn up a specification of requirements?
5. Would you consider it useful to consult with other local authorities in respect of common requirements in data processing?
6. If you answered 'yes' to Question 5, what areas do you consider the various authorities would have in common?
7. Which of the following modes of operation would you prefer should it be decided to proceed with ADP?
   • Use a consultant for systems work and programming?
   • Use a service provided regionally by other authorities?
   • Use separate and compatible equipment and share software costs with other authorities?
8. Could you indicate, if appropriate, why you would prefer to retain a separate data processing facility?
9. Do you consider the present arrangements efficient?
10. Should computerisation proceed, or if it has proceeded, what money savings have been achieved or are anticipated?
11. Where do you obtain information about commercially available computer systems?
    • From sales literature?
    • From salesmen?
    • From 'computer' associations?
    • From business associations?
    • From local government training courses?
    • From courses conducted by manufacturers?
    • From user's groups?
    • By ad hoc means?
12. From which sources would you recruit ADP staff (if required) from this region?
    • Recruit inside organisation.
    • Recruit inside and have manufacturer train.
    • Recruit outside organisation.
    • Recruit from university graduates.
    • Train operators and data processing staff using internal courses as distinct from manufacturers' courses.
    • Recruit 'experts' from outside organisation.
    • There is no expertise within the organisation.
    • The question is inappropriate.
13. Was a cost benefit analysis regarding computerisation ever undertaken?
14. In your opinion, is your authority 'on side' regarding computers?
15. All shires collect rates. Why not club together?
16. Should the Geelong Regional Commission proceed with an investigation of the benefits of a centralised word processing system?
   What had to be established was first: were there any common applications? The answer is obvious — the data processing involved in rating and voters' rolls was common to the nine councils surveyed, and that of payroll, job costing, general accounting to all (twelve) authorities. Engineering and inventory processing were common to eleven.
   In addition, eleven of the authorities used computers in one way or another (see Table 1).
As can be seen, the authorities had chosen a variety of ways of accomplishing their data processing requirements. The larger councils had obviously not accepted the idea of rate and voters' roll processing by the Local Authorities Superannuation Board computer. This idea of the LSAB acting as a service bureau was seriously considered about ten years ago.

Plans for future use of computers seemed to be fairly unclear at this stage. Authorities were asked about their plans for the mid-eighties and, as Table 2 shows, were not able to specify what their future uses would be, except in the case of the larger authorities.

Regarding the question of whether the twelve authorities had ever consulted with each other formally or informally, about their common data processing requirements; most of them (eleven) thought it useful to consult with others and had already done so. The eleven had then agreed that rating systems were common to all of them, and voters' rolls common to nine. The opinion was also shared among ten that present arrangements were not efficient in general, with two "possibly inefficient" responses included in this result.

It would appear then, that the stage was set for a positive response to the idea of clubbing together to pool computer resources. Indeed, most of the authorities agreed that this would be a useful development. Eight were in favour, four against. Computers had produced benefits in the past, primarily in reducing staff (two responses) or not increasing it (ten). Councils were able to rate earlier (five responses) and could give "better service" (four). (These benefits, incidentally, were provided by the interviewees, not the interviewer.) Asked whether their authorities supported computers, six replied that their respective bodies were "on side"; one that councillors appreciated them, although they knew nothing about them. Five however responded to this in a qualified manner claiming that computers were oversold, or there was a role for people, or that their authority was "offside" about computers.

The reader may be further confused by the reaction to another question: whether the GRC should investigate the possibility of installing a centralised word processing system. Six of the authorities opposed any investigation (let alone acquisition), one more was not prepared to comment, and five were in favour. The word "centralised" was deliberately used in the phrasing of the question. One might ask whether a "decentralised" word processing system question might have gained the majority.

We have the answer to that in a sense. At a meeting of the GRC held in mid-1979, attended by representatives of the various authorities, each representative having voting rights, a recommendation was debated that a steering committee, consisting of at least one representative from each authority be formed. This committee was to investigate whether a powerful interactive processor would be more beneficial than the current arrangements, and, in the event of a positive response, organise a cost benefit study. The recommendation was discussed for some time and it became fairly clear that there was some variations of opinion among the representatives.

The commissioned study was later accepted at the same meeting and the GRC resolved to consider the question of a steering committee subsequently.

This matter was debated in August 1979. Not all authorities were represented at this meeting and the delegates could not decide whether to set up the steering committee. It seemed that it would be necessary for their authorities to decide. Even so, the co-operative use of facilities would be long term — about six years because some councils had already installed equipment and the normal life of that would be an average of five years.

Besides, the range of operations between authorities was wide.

It took eight months for responses to reach the stage of further consideration by the GRC on 1 May 1980. Six authorities considered that they were interested in investigating the possibility of a co-operative approach, one considered that its short to medium term provision was adequate and therefore did not wish to participate, one wished to be kept informed, one had interest in membership some time in the future, and one did not wish to be included. Two failed to respond.

Two weeks later, Corio announced that it would withdraw from the steering committee as it now wished to join another study group. South Barwon also had joined this group and its representatives had been present at the May meetings as observers.

It was clear by now that a steering committee would not be formed as the authorities were not enthusiastic enough to commence a study. The reasons given were that commitments had been made and that there were diverse computer needs.

The GRC then expressed interest in the other study group. This had, in fact, been instigated by the Shire of Werribee which had complained that it had not been able to find a software system to meet its needs. The LASB had agreed with ICL Ltd to jointly fund a Property and Lands System at a cost of $1.8 million. A number of councils had shown interest and the LASB had formed a working group of various authorities.

The LASB system was envisaged as an integrated database including digitised maps and terrain and vegetation analysis. Some programs were to become available by October 1980, and the full system within 18 months. This system would be hired by the region at a cost of $125,000 per annum. All councils would be expected to participate, each having a terminal on line to the service which might well be physically located in GRC offices or some other suitable central point, but the LASB would provide personnel for the management and operation of the service. No doubt the scheme will herald a fresh round of debate.

What was the cause of the original disagreement? On the face of it, it would not seem irrational to investigate the possibility of resource pooling; on the contrary, it would seem to be the obvious course of action. Perhaps then, some authorities considered that the GRC was going beyond its legal brief in attempting to initiate such a scheme.

I have quoted from the Act establishing the GRC. It might be useful to consider that Act in relation to the Local Government Act of 1958. The latter is long and detailed and, in general terms, deals with the regulation of the proper business of local government. It is concerned with, amongst other things, the definition of the ordinary business of council in particular:

a) Superintending the conduct of the officers of the council;

b) Inquiring into the conduct of contractors or other persons employed by the council to execute any works;

c) Inquiring into the state and progress of such works;
### TABLE 1. DATA PROCESSING BY AUTHORITY

<table>
<thead>
<tr>
<th>Authority</th>
<th>Application</th>
<th>Processing</th>
<th>Authority</th>
<th>Application</th>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shire of Bannockburn (1,500 rating assessments)</td>
<td>Rating</td>
<td>SMS</td>
<td>City of Geelong West (6,524 assessments)</td>
<td>Rating</td>
<td>LASB</td>
</tr>
<tr>
<td></td>
<td>Voters' Rolls</td>
<td>SMS</td>
<td></td>
<td>Voters' Rolls</td>
<td>LASB</td>
</tr>
<tr>
<td></td>
<td>Engineering</td>
<td>Burroughs L 4000</td>
<td></td>
<td>Engineering</td>
<td>Accounting machine</td>
</tr>
<tr>
<td></td>
<td>Inventory</td>
<td>Manual</td>
<td></td>
<td>Costing</td>
<td>Accounting machine</td>
</tr>
<tr>
<td>Shire of Barrabool (6,040 assessments)</td>
<td>Rating</td>
<td>SMS</td>
<td>City of Newtown (4,400 assessments)</td>
<td>Rating</td>
<td>LASB</td>
</tr>
<tr>
<td></td>
<td>Voters' Rolls</td>
<td>SMS</td>
<td></td>
<td>Voters' Rolls</td>
<td>LASB</td>
</tr>
<tr>
<td></td>
<td>Payroll</td>
<td>L 4000</td>
<td></td>
<td>Payroll</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Costing</td>
<td>L 4000</td>
<td></td>
<td>Costing</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Accounting</td>
<td>Manual</td>
<td></td>
<td>Accounting</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Inventory</td>
<td>L 4000</td>
<td></td>
<td>Inventory</td>
<td>Manual</td>
</tr>
<tr>
<td>Shire of Bellarine (19,647 assessments)</td>
<td>Rating</td>
<td>SMS</td>
<td>Borough of Queenscliff (21,000 assessments)</td>
<td>Rating</td>
<td>LASB</td>
</tr>
<tr>
<td></td>
<td>Voters' Rolls</td>
<td>SMS</td>
<td></td>
<td>Voters' Rolls</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Quarterly</td>
<td></td>
<td></td>
<td>Payroll</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Installment Accounts</td>
<td></td>
<td></td>
<td>Engineering</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Engineering</td>
<td></td>
<td></td>
<td>Costing</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Payroll</td>
<td></td>
<td></td>
<td>Accounting</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Costing</td>
<td></td>
<td></td>
<td>Inventory</td>
<td>Manual</td>
</tr>
<tr>
<td>Shire of Corio (20,000 assessments)</td>
<td>Rating</td>
<td>LASB</td>
<td>Geelong Water and Sewerage Trust</td>
<td>Rating</td>
<td>IBM System 3-12</td>
</tr>
<tr>
<td></td>
<td>Voters' Rolls</td>
<td>LASB</td>
<td></td>
<td>Payroll</td>
<td>IBM 3-12</td>
</tr>
<tr>
<td></td>
<td>Payroll</td>
<td>Burroughs B 700</td>
<td></td>
<td>Engineering</td>
<td>IBM 3-12</td>
</tr>
<tr>
<td></td>
<td>Engineering</td>
<td>B 700</td>
<td></td>
<td>Costing</td>
<td>IBM 3-12</td>
</tr>
<tr>
<td></td>
<td>Costing</td>
<td>B 700</td>
<td></td>
<td>Inscribed Stocks</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Accounting</td>
<td>B 700</td>
<td></td>
<td>Accounting</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td>Inventory</td>
<td></td>
<td></td>
<td>Inventory</td>
<td>Manual</td>
</tr>
<tr>
<td>City of South Barwon (14,296 assessments)</td>
<td>Rating</td>
<td>LASB (Partly manual)</td>
<td>Shire of Bellarine</td>
<td>Yes</td>
<td>Up-grade computer use. Install October 1980</td>
</tr>
<tr>
<td></td>
<td>Voters' Rolls</td>
<td>LASB</td>
<td></td>
<td>Payroll</td>
<td>Nixdorf VRC</td>
</tr>
<tr>
<td></td>
<td>Payroll</td>
<td>Nixdorf VRC</td>
<td></td>
<td>Engineering</td>
<td>Nixdorf VRC</td>
</tr>
<tr>
<td></td>
<td>Engineering</td>
<td>Manual</td>
<td></td>
<td>Costing</td>
<td>Nixdorf VRC</td>
</tr>
<tr>
<td></td>
<td>Costing</td>
<td>Nixdorf VRC</td>
<td></td>
<td>Accounting</td>
<td>Nixdorf VRC</td>
</tr>
<tr>
<td></td>
<td>Accounting</td>
<td>Nixdorf VRC</td>
<td></td>
<td>Inventory</td>
<td>Nixdorf VRC</td>
</tr>
<tr>
<td></td>
<td>Inventory</td>
<td>Nixdorf VRC</td>
<td></td>
<td>Private Streets</td>
<td>Nixdorf VRC</td>
</tr>
<tr>
<td>Shire of Bannockburn</td>
<td>Possibly</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shire of Barrabool</td>
<td>Yes</td>
<td>Have been drawn up for an in-house mini. This will be the accounting function, perhaps rating, on-line engineering, etc. Went to tender. Saw several installations.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shire of Bellarine</td>
<td>Not in short term.</td>
<td>Have good idea of long term requirements. Long term means less than five years.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shire of Corio</td>
<td>Yes</td>
<td>Full application including engineering.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Borough of Queenscliff</td>
<td>Not apart from LASB.</td>
<td>No.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City of Geelong</td>
<td>No.</td>
<td>No.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City of Geelong West</td>
<td>No.</td>
<td>No.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City of Newtown</td>
<td>No.</td>
<td>No.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City of South Barwon</td>
<td>Yes.</td>
<td>No definite plans as yet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geelong Water and Sewerage Trust</td>
<td>Yes.</td>
<td>Full application including engineering.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
d) Generally giving such directions from time to time as may be necessary for carrying into effect the purposes of this Act. (Acts 1958)

Item d) is concerned with planning, hence resource allocation, but within the context of the Act, relates, I believe, to the district over which a council has responsibility. It is thus proper for a council to plan for and allocate resources to a computer facility, or to employ a bureau, even if that bureau were controlled by another local government authority. It would be improper for a council to plan for a computer facility for another council without the consent of the latter council. The foregoing is to be read in the context of ordinary business.

The Act also provides for extraordinary business, that is: “any business other than ordinary business”; it gives the Minister the power, whenever he sees fit, to call a council meeting, and Clause 189 describes the power a council may be given by a “special order”. None of these special powers implicitly give any council the power to plan for another. Presumably one council could be so empowered.

The GRC is not a council. It was created by an Act of 1977 and its brief is to plan for the Geelong region. This is more nebulous than the very specific Part VII of the 1958 Act which set out the purposes for which a council may make by-laws. The GRC can plan perhaps, but it cannot allocate the resources of a particular council.

This seems to be the root of the problem. The local councils say they are worried that the GRC is trying to “take over”. This it cannot do. In fact the sensibilities of councils were taken into account when the Act establishing the GRC was passed since the powers of the Commission were reduced by a series of amendments during the passage.

There is no need to explore further the politics of local government in the Geelong area except to remark that, politics have made the process of decision-making difficult. Now, probably many people who work with computers dislike politics, particularly when decisions are being made to purchase hardware or whether to adopt a new technology. The way to solve the problem of politics is to place decision-making in the hands of a forceful individual. In that case, negative decisions are rare and there is no guarantee that the positive decisions are at the same time rational ones.

Another way is to let time go by. This is, in effect, what has happened. The matter of technology was discussed by local councils about 1970. They were, at that time, considering changing over from manual systems. A seminar was held. The members adopted a negative attitude then. But as time went by, computer technology was adopted anyway, but in some cases, without the benefit of a reasoned cost analysis.

What can be inferred from this case? From the point of view of an EDP consultant, it might be considered an exercise that was less than successful. Which in turn implies that to an EDP consultant, “success” is normally translated into decisions to acquire hardware or to implement new technology, certainly not to do nothing.

It also illustrates that there is a class of problem which may be susceptible to computer technology, but which cannot be solved in the short term. It is suggested that these problems are typically fairly complex ones, under the control of committees and are difficult to solve when members have aims which are in conflict with the aims of the committee. In that case the traditional cost benefit analysis approach to decision-making may not be used.

REFERENCES
ATTITUDES TOWARDS COMPUTERS: A SURVEY OF DIVERSE PERSONS

A.G. Smith*

*The author is with the Department of Education, University of Newcastle, Newcastle, NSW, 2308. Manuscript received 20 February 1981, and revised 27 April 1981.

A 40 item Likert attitude instrument, measuring attitude towards computers in society, was constructed and administered to a sample of 414 people in Newcastle, New South Wales. The instrument is shown to discriminate people on the basis of attitude towards computers. The nature of a positive versus a negative attitude is examined. Respondents were further studied in five groups, from keen computer users to strongly negative opponents, and some implications of their attitudes for education programmes and for further research are discussed.

Keywords and phrases: Attitudes towards computers, public image, social implications.
CR categories: 1.5, 2.1, 3.3, 3.4.

INTRODUCTION

The period since 1977 has seen a heightened awareness in Australia of the impact of computer technology on society in general, and on the work place in particular (Goldsworthy, 1979; Frankel and Maynes, 1979). The purpose of this paper is to report the results of a preliminary survey of attitudes of the public towards computers.

The period since 1977, public debate over the impact of computers on society was stimulated by a major strike by Telecom technicians, several important television programmes, and articles such as the "computer holocaust" series in The Australian (September 1978). At a time of economic difficulty and high unemployment, these contributions stressed a negative impact of computers on work. Further industrial campaigns by Telecom unions, journalists and bank tellers have reinforced this negative view.

The question may be asked, however, whether this negative view of computers in society is shared by the public. There has been a dearth of empirical research into community perceptions of the role of computers in society. No Australian work on the problem was located.

Lee (1970) and Gittlelbe and Borodin (1973) reported several early surveys of attitude towards computers in the United States. Basically, earlier surveys suggest a positive-negative continuum of attitude towards computers. On the negative side, people reportedly have been concerned about computers and unemployment, depersonalisation, privacy, computer error, and the misuse of information stored by computer. On the positive side, computers have been viewed as beneficial in health, education, space exploration and many other areas of society.

Aims

The investigation reported here was conducted as a pilot study with the following aims:

1. To construct and administer to a moderately large sample of respondents an attitude instrument surveying the major concerns discernible from previous research and from public debate;
2. To analyse the instrument in order to determine which attitude constructs, if any, deserve closer attention in further research; and
3. To determine whether subgroups possessing different attitude profiles could be defined within the sample.

RESEARCH METHOD

As many statements as possible of attitude towards computers and their use in society were gathered using the works of Lee (1970), Martin and Norman (1973) and Packard (1978) as a starting point. After a priori examination, 40 statements were selected which covered a wide range of feeling about computers. The statements were randomly assigned in a six-point Likert attitude scaling instrument (Nunnally, 1967). Choice points were labelled from "strongly agree" to "strongly disagree", with two central choices labelled "not sure but tend to agree" and "not sure but tend to disagree". To the extent that a neutral "don't know or undecided" category was not provided, the instrument was slightly forced, but this strategy discourages response styles (Nunnally, 1967, p. 522). Each attitude statement, to which a respondent replied in one of the six categories of agreement or disagreement, constituted an item in the attitude scale. Unanswered items were recorded as missing data and all available data included in the statistical analyses. Background data included sex of respondent, age level (in five categories), level of education (seven categories), amount of computer experience (four points, often to never), and area of work or study (from 16 delineated). Responses to the survey were anonymous.

Respondents

A large sample of convenience (N=414) was obtained. The survey administration took place in the summer of 1978/79. Respondents included students and staff at Newcastle's university, college of advanced education, technical college, a large hospital including some patients, non-academic professional and business people, housewives, and a group of unemployed persons.

The data were analysed with the aid of Youngman's (1976) suite of statistical programs and Dixon and Brown's (1979) statistical package (BMDO).

Analysis of the survey was carried out in three parts. First, the matrix of answers to the 40 attitude statements by respondents (40 x 414) was analysed. This analysis was necessary to answer questions related to the quality of the attitude scale. Second, the answers and background data provided by respondents were analysed in order to obtain an attitude profile for the sample as a whole. Lastly, a cluster analysis was performed on the matrix in order to discover any attitudinal subgroups in the sample.

RESULTS

Analysis of the Instrument

The instrument was analysed in order to determine its statistical reliability, and whether any useful subscales (attitude themes) existed within the instrument. Attitude items were scored in the direction of a positive attitude towards computers. The alpha co-efficient of internal consistency reliability was calculated at 0.80 for the total scale. The product-moment correlation matrix for the 40 items was computed and factor analysed by the method of principal components analysis followed by Little Jiffy Mark IV oblique rotation (Youngman, 1976). A three factor rotated solution was accepted as defining the most useful subscales within the instrument. Subscales of 13, 14 and 10 items were produced. Several items loaded on two factors, and one item loaded on all three. Of the original 40 items, 28 loaded on one or more of the three subscales. The item contents of the three subscales are presented in Tables 1, 2 and 3 respectively. Reliabilities for the three subscales were...
the items in this factor demonstrate an appreciation of the encouraging, bearing in mind the exploratory nature of the 0.87, 0.71 and 0.60 respectively. Overall reliability for the computers" perspective. Some misconceptions about com­
tuators, or at least that they are held in as 14, 27 and 28, suggest that a certain omnipotence is association is not necessarily realistic. Some of the items, such 39, computers ... hardship for the working man. 6 the society with a strong computer concern 28 retained items was 0.76. All these co-efficients are most 28 computers are represented in this factor, but basically the items puters and anxiety about them is strongly suggested by items such as 5, 10 and 28. The negatively loading item 9 
ments (humans) ... surpassed by ... intelligent computers. Strongly disagree 2.10 1.40 .43 Disagree 2.90 1.32 .38 Disagree 2.22 1.27 .30 (bimodal) 3.67 1.51 -.34
TABLE 1. Salient Items and Sample Responses, First Oblique Factor Subscale.

<table>
<thead>
<tr>
<th>Item</th>
<th>Item Content</th>
<th>Modal Response</th>
<th>Sample Mean</th>
<th>Standard Deviation</th>
<th>Oblique Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Some day ...(computers)... running our lives...</td>
<td>Disagree</td>
<td>3.03</td>
<td>1.71</td>
<td>.75</td>
</tr>
<tr>
<td>4</td>
<td>Man's freedom ... jeopardised ... increased use ... computers.</td>
<td>Disagree</td>
<td>3.05</td>
<td>1.68</td>
<td>.67</td>
</tr>
<tr>
<td>15</td>
<td>... individual ... will not count for much anymore.</td>
<td>Strongly disagree</td>
<td>2.59</td>
<td>1.65</td>
<td>.63</td>
</tr>
<tr>
<td>33</td>
<td>Computers ... throw us out of work.</td>
<td>Disagree</td>
<td>3.00</td>
<td>1.53</td>
<td>.63</td>
</tr>
<tr>
<td>21</td>
<td>... personal privacy will decrease.</td>
<td>Agree</td>
<td>3.94</td>
<td>1.50</td>
<td>.55</td>
</tr>
<tr>
<td>12</td>
<td>Business is ... too dependent on (computers).</td>
<td>(bimodal)</td>
<td>3.62</td>
<td>1.66</td>
<td>.46</td>
</tr>
<tr>
<td>10</td>
<td>computers ... know too much.</td>
<td>Disagree</td>
<td>2.62</td>
<td>1.62</td>
<td>.43</td>
</tr>
<tr>
<td>39</td>
<td>computers ... hardship for the working man.</td>
<td>Disagree</td>
<td>2.66</td>
<td>1.50</td>
<td>.45</td>
</tr>
<tr>
<td>6</td>
<td>... computerisation ... loss of ... jobs.</td>
<td>Agree</td>
<td>4.46</td>
<td>1.50</td>
<td>.43</td>
</tr>
<tr>
<td>28</td>
<td>... humans ... will be surpassed ... by ... intelligent computers.</td>
<td>Strongly disagree</td>
<td>2.19</td>
<td>1.40</td>
<td>.43</td>
</tr>
<tr>
<td>17</td>
<td>(Society ...) better off without computers.</td>
<td>Disagree</td>
<td>2.90</td>
<td>1.32</td>
<td>.38</td>
</tr>
<tr>
<td>-9</td>
<td>Computers should be used as much as possible.</td>
<td>Disagree</td>
<td>2.22</td>
<td>1.27</td>
<td>.30</td>
</tr>
</tbody>
</table>

TABLE 2. Salient Items and Sample Responses, Second Oblique Factor Subscale.

<table>
<thead>
<tr>
<th>Item</th>
<th>Item Content</th>
<th>Modal Response</th>
<th>Sample Mean</th>
<th>Standard Deviation</th>
<th>Oblique Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Computer ... improve ... teaching in the future.</td>
<td>Tend to agree</td>
<td>3.94</td>
<td>1.41</td>
<td>.57</td>
</tr>
<tr>
<td>16</td>
<td>Computer Assisted instruction ... children learn more efficiently.</td>
<td>Tend to agree</td>
<td>3.53</td>
<td>1.47</td>
<td>.57</td>
</tr>
<tr>
<td>20</td>
<td>Computers ... make life easier for all.</td>
<td>Agree</td>
<td>3.99</td>
<td>1.35</td>
<td>.46</td>
</tr>
<tr>
<td>13</td>
<td>... necessary ... handle ... complex problems ... modern society.</td>
<td>Agree</td>
<td>3.81</td>
<td>1.58</td>
<td>.45</td>
</tr>
<tr>
<td>26</td>
<td>... will be capable to diagnosing all of man's diseases.</td>
<td>Tend to agree</td>
<td>3.19</td>
<td>1.50</td>
<td>.42</td>
</tr>
<tr>
<td>14</td>
<td>... can think better than human beings.</td>
<td>Strongly disagree</td>
<td>2.05</td>
<td>1.38</td>
<td>.39</td>
</tr>
<tr>
<td>28</td>
<td>Eventually ... (humans) ... surpassed by ... intelligent computers.</td>
<td>Strongly disagree</td>
<td>2.10</td>
<td>1.40</td>
<td>.36</td>
</tr>
<tr>
<td>10</td>
<td>Data banks (of personal information) ... are desirable.</td>
<td>Strongly disagree</td>
<td>2.42</td>
<td>1.50</td>
<td>.35</td>
</tr>
<tr>
<td>27</td>
<td>Computers ... trustworthy and secure.</td>
<td>(bimodal)</td>
<td>3.54</td>
<td>1.53</td>
<td>.33</td>
</tr>
<tr>
<td>27</td>
<td>A computer can learn.</td>
<td>Strongly disagree</td>
<td>2.89</td>
<td>1.64</td>
<td>.32</td>
</tr>
<tr>
<td>36</td>
<td>Knowledge ... computers should be taught in schools.</td>
<td>Agree</td>
<td>4.72</td>
<td>1.27</td>
<td>.32</td>
</tr>
<tr>
<td>9</td>
<td>... should be used as much as possible</td>
<td>(bimodal)</td>
<td>3.67</td>
<td>1.51</td>
<td>.30</td>
</tr>
<tr>
<td>-29</td>
<td>... more (computers) ... will not improve society,</td>
<td>Disagree</td>
<td>3.14</td>
<td>1.53</td>
<td>.37</td>
</tr>
<tr>
<td>-17</td>
<td>... better off without computers.</td>
<td>Disagree</td>
<td>2.22</td>
<td>1.27</td>
<td>.34</td>
</tr>
</tbody>
</table>

The three subscales contain many separate ideas, and the analysis reveals two important points about them. Firstly, it shows that various expressed opinions on the effect of computers on work, individual freedoms, privacy and other issues in fact are closely related to one another in peoples' minds. The many separate ideas are different aspects of a more global concern over computers in society.

Secondly, the analysis suggests that a positive-negative continuum of attitude towards computers does exist. The positive side of the scale is represented by subscale 2. The negative side of the scale is more complex, being composed of the general concern subscale (1), and the more irrational threat subscale (3). Subscales 1 and 3 are positively intercorrelated (0.63), and both are negatively correlated with subscale 2 (−0.50 in each case), which verifies the interpretation.

**Sample Characteristics**
Tables 4, 5 and 6 summarise the background information pertaining to respondents. Some key points to be noted from these tables are that:

- **a)** Computer experience was largely restricted to the under 29 age group;
- **b)** Not many computer professionals were sampled;
- **c)** University students and graduates are much more likely than others to have computer experience; and
- **d)** Men were much more likely than women to have used a computer.

For the total sample, 40 percent had never acquired direct experience of computers. Only 9.4 percent of the total sample frequently used a computer, and only an additional 26.5 percent sometimes did so. On average, the sample was characterised by little knowledge of or contact with computers, and only a small proportion could be said to be at all well-informed about computers. It is therefore particularly interesting that the sample rejected the overall
negative "flavour" of the survey instrument. On average, respondents were balanced in their attitude towards computers. The sample mean score over all 40 items was 123/240 (51.3 percent). For the two negative attitude subscales, the sample averaged 47.7 percent or a "tend to disagree" position. In the case of the positive attitude subscale, however, respondents averaged 57.7 percent or "tend to agree". For the model respondent, computers had a basically positive role to play in society, but there were a number of negative aspects to their social impact. In particular, he was concerned about the impact of computers on job opportunities. Other items with which he strongly agreed were that (a) a strike in the computer industry could cause national chaos; (b) that computers could be used "for evil" in the "wrong hands"; and (c) that knowledge about computers should be taught in schools. Items most strongly disagreed with were (a) that computers think better than human beings; (b) that eventually humans will be surpassed by intelligent computers; (c) that if computers put one out of work, one should accept less pay; and (d) that we would be better off without computers.

TABLE 6. Main Field of Endeavour by Sex and Frequency of Computer Use.

<table>
<thead>
<tr>
<th>Main Field of Endeavour</th>
<th>Sex*</th>
<th>Computer Use</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>F</td>
<td>Often</td>
<td>Sometimes</td>
<td>Rarely</td>
<td>Never</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commerce, clerical business</td>
<td>58</td>
<td>27</td>
<td>7</td>
<td>32</td>
<td>29</td>
<td>18</td>
<td>86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineering</td>
<td>90</td>
<td>1</td>
<td>9</td>
<td>26</td>
<td>26</td>
<td>30</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agriculture, building, other commercial</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Psychology, medicine, nursing</td>
<td>4</td>
<td>29</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>18</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Science</td>
<td>21</td>
<td>14</td>
<td>4</td>
<td>12</td>
<td>8</td>
<td>11</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maths</td>
<td>8</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>21</td>
<td>33</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>34</td>
<td>52 (2)*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture, English/History teaching</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>11 (1)*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Science</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other incl. unemployed</td>
<td>21</td>
<td>43</td>
<td>1</td>
<td>7</td>
<td>13</td>
<td>43</td>
<td>64 (2)*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>M 245</td>
<td>F 165</td>
<td>39</td>
<td>110</td>
<td>96</td>
<td>165</td>
<td>409 (5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: * Figures in parentheses represent frequencies of respondents not supplying relevant data.

Taxonomical Analysis of the Sample by Attitude

The cluster analysis was undertaken to determine whether any groups of people would emerge who differed both in attitude towards computers and in background personal characteristics. The background variables were standardised and included in a taxonomic analysis with standardised responses to the attitude items. All items were used in the analysis to maximise discrimination. Commencing from a random allocation of respondents to nine groups, relocation analysis and 'error sum of squares' clustering produced five reasonably homogeneous subgroups in the sample. A multiple discriminant analysis strongly supported these groupings; only two items in the analysis failed to contribute significantly to the discrimination.

Table 7 presents the group raw score means for the six background variables, and the 28 subscale attitude items described in Tables 1 to 3. If these figures are graphed by attitude subscale, using the sample mean as a reference point, some most interesting differences emerge between the groups.

Group 1 might be called the “mature age, technical trades” group. Members of the group were older than the average, they were the least well-educated group, and they rarely used a computer in their work. Despite these formal “deficits”, however, the group held no extreme views about computers in society, and their responses to most items were close to the sample average. For this group, computers had a basically positive role in society, although there were some important negative effects on jobs.

Group 2 is a most interesting group. This is the “computer enthusiast” group. Group members used a computer often. The group had a strongly positive attitude towards computers, and the group strongly rejected all negatively stated items. For this group, computers were unambiguously good for society and the individual.

Group 3 was the youngest group, and members were well-educated. This was substantially a group of university and college students, who, as part of their training, had some experience of computers. Perhaps because they knew something about many of the issues raised in the survey as well as something about computers, the group had a complex profile which expressed both good and bad aspects to the role of computers in society.

Groups 4 and 5 are particularly interesting groups, especially when contrasted with Group 2. Group 4 was a group of people with mixed backgrounds who held a negative attitude towards computers. A significant feature of this group is the fact that members had, on average, rarely used a computer. Group 5 was characterised by very well-educated people who had almost no experience of computers. Group 5 held strong anti-computer attitudes which were virtually a mirror image to those expressed by the “computer enthusiasts” group. There is a sex difference between Groups 2 and 5, but the more important difference lies in the area of professional qualifications. Group 2 was substantially a group of males qualified in science, engineering, and commerce. Group 5 was substantially a group of females qualified in the arts, humanities, and caring professions.

DISCUSSION

The results of the attitude survey are of interest for three main reasons. Firstly, the instrument was shown to sample several attitude dimensions which reliably discriminated among peoples’ views of computers. The instru-
TABLE 7. Raw-score Group and Sample Means on Selected Items.

<table>
<thead>
<tr>
<th>Item</th>
<th>Range of Values</th>
<th>Sample Mean</th>
<th>GROUP means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 (N=104) 2 (N=103) 3 (N=83) 4 (N=70) 5 (N=54)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Sex 1 (M) — 2 (F)</td>
<td></td>
<td>1.41</td>
<td>1.33</td>
</tr>
<tr>
<td>Age 1 - 8</td>
<td></td>
<td>3.05</td>
<td>3.49</td>
</tr>
<tr>
<td>Post-secondary Education 1 - 5</td>
<td></td>
<td>3.37</td>
<td>3.04</td>
</tr>
<tr>
<td>Computer Experience 1 (of) - 4 (Nvr)</td>
<td>1 - 6 (SD) (SA)</td>
<td>2.92</td>
<td>3.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>3.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>3.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>4.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>3.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>2.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>2.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>3.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>2.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>33</td>
<td>3.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37</td>
<td>2.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>39</td>
<td>2.66</td>
</tr>
</tbody>
</table>

The results enable one to offer a tentative answer to the basic question of what is meant by either a positive or a negative or a balanced view of computers and their role in society. A person with a highly positive attitude towards computers is likely to be male, under 30 years of age, relatively highly experienced with computers, and either studying or working in commerce, selected clerical occupations, or certain physical science areas such as engineering. The highly positive person likes to see computers used as much as possible; computers are good for society in all major respects, and hold no threat or negative implications of importance for society.

The highly negative person, on the other hand, is likely to be a female around 30 years of age, very well-educated and with a strong orientation towards the arts or humanities professions. The strongly negative person, by virtue of their vocational orientation, is unlikely to have any direct experience in the use of a computer, is alienated by computers and is very concerned over effects on people at both an individual and at a social level.

A balanced view of computers in society was reflected in the sample as a whole. Computers were seen as being both indispensable to and desirable for modern society and its living standards, but a strong moderating influence was reflected in concerns over the potential impact of computers on work opportunities in particular.

It is suggested by the data that there is a multivariate relationship among background variables such that attitude towards computers is substantially a function of one's vocational orientation. The five clusters of people which emerged were rank ordered, from most favourable towards computers to least favourable, on the basis of frequency of computer use, and on the basis of post-secondary educational qualifications. It would also appear that there is a sex interaction with computer experience and the likelihood of holding a negative attitude. These interactions are not simple, however, in that the most positive group was not the best educated, but the most negative group was exceptionally well-educated. Similarly, the least well-educated and oldest group, which might have been expected to be
negative in attitude because of computer effects on the work place, was in fact balanced in its profile.

These observations suggest the existence of a “technological orientation”: the degree of technicality in a field of work versus orientation towards people and the caring professions. This technological orientation may explain enthusiasm or lack of it for computers, and may explain the sex difference observed between groups insofar as sex is related to area of occupational choice. Such an hypothesis would appear to be supported by the work of Goldman, Platt and Kaplan (1973).

The question is also raised as to whether a negative attitude or even fear of computers could be ameliorated through non-threatening educational experiences of computers. It is interesting to note that the more positively disposed groups in the sample believed that knowledge about computers should be taught in schools and that computers have a positive role to play in improving the effectiveness of education in the future, whereas the negatively disposed groups rejected both these opinions. One implication of this may be that a multidisciplinary form of computing should be taught at school level as well as tertiary level, and that such teaching should not be restricted to the technical science subjects.

ACKNOWLEDGEMENT

The work of Ellen M. Bray, H. Schultz, and A.G. Williams in the original construction and administration of the survey instrument is gratefully acknowledged. Without their work, the sample of respondents would not have been as large nor as diversified as finally emerged.

REFERENCES


ASYM: A DIVISIVE MONOTHETIC CLASSIFICATORY PROGRAM FOR BINARY DATA

D.A. Hedges† and W.T. Williams††

†CSIRO, Division of Animal Production, Pastoral Research Laboratory, Armidale, NSW, 2350. ††Australian Institute of Marine Science, Private Bag No. 3, Townsville, Qld., 4810. Manuscript received 19 November 1980, and revised 22 December 1980.

A program for the monothetic divisive classification of binary data using the asymmetric Jaccard measure is described. The new program, ASYM, contains a sorting strategy based on group heterogeneity which is not group-size dependent but instead is particularly sensitive to species diversity. Comparisons with the group-size dependent program DIVINF indicate that for those ecological applications where species diversity is important ASYM can be expected to give the more useful result.

Keywords and phrases: numerical classification, ecological survey.

CR categories: 3.12, 5.19.

INTRODUCTION

Divisive monothetic algorithms are widely used for the classification of binary data. They have proved particularly effective in ecological survey where the data are typically, though not invariably, the presence or absence of a number of species in sample sites over a defined study area. Such algorithms have considerable computational advantage over agglomerative methods; and, indeed for very large data sets they remain the only computationally feasible methods available.

As far as we are aware all divisive monothetic computer programs in common use are based on similarity measures which are symmetric (i.e. both joint presences and joint absences count towards similarity). This is certainly true of DIVINF (Lance and Williams 1968) which, until recently, was the only such program available in the TAXON library of the Cyber 76 computer in Canberra.

In a recent paper Williams and Bunt (1980) pointed out that situations exist in ecology when measures which ignore double-zero matches may be preferred and presented algorithms for a monothetic divisive program based on the asymmetric Jaccard measure. These algorithms have now been incorporated into a fully-automatic FORTRAN program called ASYM after their original ad hoc version from which it evolved. The program is available in the TRYNTAX library on the Cyber 76 computer in Canberra. Detailed instructions for accessing and using it are available in the TAXON User's Manual (CSIRO, Division of Computing Research User's Manual No. 6). Here we wish only to present brief details of the algorithms and some comments on our experience to date with their use. Further, although many other strategies and measures (both symmetric and asymmetric) could be included in the program the rigorous assessment of the relative merits of these alternatives is beyond our current resources. Our prime concern has been to provide an algorithm based on an asymmetric measure and a program which others may extend should they so desire.

ALGORITHMS

Measures

Two measures are currently available though others are under consideration. In the conventional a, b, c, d symbolism of the 2 x 2 contingency table for the presence/absence of the jth and kth 'effective' species the two measures are:

i) the asymmetric Jaccard measure

\[ S_{jk} = \frac{a}{a + b + c} \]

ii) the symmetric simple matching coefficient

\[ S_{jk} = \frac{a + d}{a + b + c + d} \]

An effective species is one which is not everywhere present or absent in the group under consideration. For each of the ne effective species the quantity

\[ B_j = \sum_{k \neq j} S_{jk} \]

(i.e. the row or column sum of the symmetric matrix \( S_{jk} \) with all zero diagonal) is calculated and the species with the largest such value is used for monothetic division.

The simple matching coefficient was included for comparative purposes. As it has not, in our admittedly rather limited experience with it, proved particularly effective, all future general references to ASYM in this paper imply the Jaccard measure.

Strategy

Two strategies for choosing the group next to be divided are available. The original method proposed by Williams and Bunt (loc. cit.) is based on a measure of group heterogeneity defined as

\[ H = n_e (n_e - 1) - \sum B_j \]

On occasions this strategy has led to an unacceptable degree of chaining down the positive side of the hierarchy. An alternative strategy based simply on group size (i.e. selects the largest remaining group with heterogeneity greater than zero for division) has been provided though little experience in its application is available. A more sophisticated alternative (e.g. weighting by some function of depth in the hierarchy, species frequency and/or group size) is required.

Ambiguities

Cases have been noted where two, or more, attributes have the same contribution to a particular division (i.e. have equal values of B). Such cases are resolved by selecting the most abundant (or the first of a set with equal highest abundance) as the division attribute.

Final Groups

A group with no effective species cannot be further divided. However, with the Jaccard measure cases may arise when a group contains n effective species, but because no quadrat in the group contains more than one of these species, a (the number of joint presences of two effective species) is everywhere zero. Although therefore, no single optimal division species exists the group must consist of n sets of identical quadrats with each set defined by the presence of one of the n effective species; it may also contain a
further identical set with none of the effective species. The program regards such a group as final (i.e. not to be further divided) and lists the identity (and the numbers of occurrences) of the n species.

Potential Division Species
Cases have also been noted where only some of the n effective species never occur with any other effective species. Such species are not considered as potential division species (they have By = 0) but again each will define a unique subset of quadrats as described above. Such subsets are not split off by the program; they will be included in the negative side of any subsequent divisions and would, if division continued, eventually become a final group as defined in the above section. The program prints out both the numbers of effective species and the numbers of potential division species (i.e. those for which B_j > 0).

DATA
The coefficients used are only defined for binary (qualitative) data. However, provision has been made for disordered multistate attributes by treating each m-state attribute as m separate binary attributes (i.e. the presence/absence of each of the m states).

Missing data values are not usually a problem in ecological survey data. Thus, while the program will accept missing values it simply treats them as though they had been recorded as zero (i.e. absent). This may lead to the misclassification of some sites for which a division species was recorded as missing.

Both individuals (samples) and attributes (species) may be masked out of the analysis and, as the data is packed (60 items per word), the program can handle quite large data sets.

COMPARISON WITH DIVINF
The two programs differ in two important respects:

i) Symmetry — as previously stated the DIVINF information statistic is symmetric (i.e. double-zero matches count towards similarity). Thus DIVINF is sensitive to both negative and positive associations between species and frequently selects division-species for their negative associations. This is not the case with ASYM as the Jaccard measure is only sensitive to positive associations (i.e. joint absences are ignored);

ii) Measure of group heterogeneity — DIVINF uses Shannon information content as its measure of group heterogeneity. This is highly group-size dependent (Williams et al. 1971) and therefore a group will tend to be chosen for division simply because it is large. Obviously the group-size algorithm of ASYM is entirely group-size dependent. However, the original ASYM measure of heterogeneity (H), which we have usually preferred is completely independent of group size. It is instead dependent on the numbers of effective species and is therefore much more sensitive to species diversity.

Williams and Bunt (loc. cit.) found this sensitivity to floristic heterogeneity particularly useful in revealing the vegetational character of their tidal forests in a way which was independent of their sampling strategy. In a study of the effects of fertilizer application to grazed pastures Taylor (1980) also found the results of ASYM more meaningful than those from DIVINF.

In the latter study the application of fertilizer changed the species composition of the pasture but it also substantially reduced the species diversity. Both programs separated the fertilized sites from the unfertilized controls. ASYM then revealed a meaningful pattern among the more floristically diverse unfertilized sites which was associated with the transfer of nutrients by the sheep grazing the pastures. The group-size dependency of DIVINF resulted in rather unprofitable dichotomies within the more numerous, but floristically impoverished, fertilized sites.

ACKNOWLEDGEMENTS
The program uses a number of generalised input subroutines from the TAXON library written by Mr D. Ross and Ms M. Thomas. We gratefully acknowledge their help with these routines and with the general development of the program. We also thank Miss Ruth Macready for her assistance in producing the detailed users guide incorporated in the TAXON Users Manual.

REFERENCES
DOTX: A NEW PROGRAM FOR THE PRODUCTION OF AN ORDERED TWO-WAY BINARY TABLE

H. J. Clay* and W. T. Williams*

*Australian Institute of Marine Science, Private Bag No. 3, Townsville, Qld. 4810.

After normal and inverse classification of a binary data-matrix, it is often desired to combine the two classifications into a "two-way table", which may then be ordered so that the most dense cells are towards the top left-hand corner. A new program, DOTX, is now available which, given the data-matrix and classificatory information, carries out the complete procedure.

Keywords and phrases: classification, two-way tables.
CR categories: 5.12, 4.40, 5.31.

INTRODUCTION

We assume a binary data-matrix, such as is common in plant ecology; in such a matrix the element in row $i$ and column $j$ is 1 or 0 according to whether the $j$th column-entity (which we shall here call an attribute) is present or absent in the $i$th row-entity (which we shall here call an individual). Alternative terminologies exist in the literature. We further assume that both the row- and column-entities have been subjected to a numerical classificatory procedure which divides the entities into mutually exclusive groups; the number of groups is usually specified in advance by the user.

We shall call a classification of the row-entities a normal classification, and a classification of the column-entities an inverse classification; the terminology is not ideal, but is now entrenched in the literature.

Macnab (1965) has pointed out that most existing classifications use "one-parameter" models. If, as is usual, major differences exist between individual row- and column-totals, a normal classification will be dominated by the row-totals, an inverse classification by the column-totals. In ecology this produces the familiar result that a normal classification largely reflects site richness, an inverse classification largely reflects species abundance. Macnab outlined a possible procedure for a divisive "two-parameter" model, based on both row and column totals, but no production program yet exists. It has therefore been usual to carry out both classifications and then combine these into a "two-way table". This is in effect a print-out of the complete data-matrix, but with the rows and columns rearranged so that all members of a classificatory group are together, and separated as a block from the members of other groups; the symbols 1 and 0 are usually replaced by symbols (in our case a capital X and a full-stop) which make the presences more visually distinctive. The block of symbols representing the presence or absence of members of a group of attributes in a group of individuals we shall call a cell; if there are $n$ groups in the normal classification, and $s$ groups in the inverse, the complete data-matrix will now be divided into $ns$ cells.

In such a table most interest lies in the "dense" cells, those with a high proportion of presences; but these are often scattered haphazardly throughout the table. It has therefore been common practice later to rearrange the table so that the densest cells are towards the top left-hand corner. In this communication we describe a program which carries out the entire procedure automatically.

BRIEF DESCRIPTION

The program is in FORTRAN; it expects a binary matrix, with all entries 1 or 0 in FORMAT(7211), since this is the conventional input for the classificatory programs of the TAXON package. The individuals are deemed to be labelled 1, 2, 3, ... in the order in which they appear as rows in the input data-matrix; the attributes are similarly labelled in the order in which they appear in the input. For each of the two prior classifications (normal and inverse) the following information is required: the number of groups into which the entities have been divided, the size of each group (i.e., the number of entities within it), and the serial numbers of the entities within that group. The program first generates an initial two-way table in which the row and column entities are sorted into their groups, and the groups are arranged in the order in which their definitions were read in; this table can be printed out if required. The program then calculates the density (as a percentage of presences) of each cell, sums the cell-densities by rows and columns, and rank-orders the totals. It then rearranges the table so that the row-groups are in descending order of cell-density row totals, and the column-groups are in descending order of cell-density column totals from left to right. The table is then printed out in this form. The cell-densities, as percentages, are printed out before and after rearrangement. If there are too many attributes to be accommodated on a single output page, the program will itself arrange for extra pages. Provision is made for names of individuals (as alphanumeric character strings) to be printed beside the row-entity serial numbers if this is required.

RESTRICTIONS

i) The data-matrix is currently restricted to 300 individuals specified by 100 attributes.

ii) The maximum permitted number of groups in the normal classification is 25, in the inverse classification 20.

iii) Individual names are restricted to 30 alphanumeric characters; they are read in under FORMAT(3A10).

iv) The input matrix can be on cards or TAPE5.

v) The individual names, if used, must be in the same order as the row-entities of the input data-matrix; they can be on cards or TAPE5.

CONTROL CARDS

All are in FORMAT(14); they are copied on to the output.

Card No.

1 NID, NATT, IFLAG, JFLAG, LUN, NAMLUN

NID = No. of individuals; NATT = No. of attributes.

IFLAG = 0 if the initial table is not required; = 1 if it is.

JFLAG = 0 if individual names are not to be read in;

= 1 if they are.

LUN = 0 if input matrix is on cards; = 1 if on file.

NAMLUN = 0 if individual names are on cards; = 5 if on file.

If the data-matrix is on cards, the data-cards now follow.

2 The number of groups in the normal classification, followed by the group-sizes (i.e., the number of individuals in each successive group).

3 This card (and continuation cards if required) contains labels identifying all the members of the first group designated, followed by labels for...
the second group, and so on.

4. As for card No. 2, but for the inverse classification.

5. As for card No. 3, but for the inverse classification.

If individual names are to be used, and are on cards, the name-cards (one 30-column name to each card) now follow.

A print-out of the program can be obtained from the senior author.

REFERENCE

REMARKS ON "A COMPARISON BETWEEN PASCAL, FORTRAN AND PL/I"

A.H.J. Sale*

*The author is with the Department of Information Science at the University of Tasmania, Hobart, Tasmania. Manuscript received 15 March 1981.

The remarks in this communication relate to a paper in the February 1981 issue of the Australian Computer Journal, by D.J. Kewley. A plea is made for more accurate titling of papers, and the usual sophistication level of Pascal compilers is discussed.

Keywords and phrases: execution times, compiler evaluation, Pascal, scientific performance.

CR categories: 3.1, 3.2, 4.6.

ACCURACY IN TITLES

It can be considered to be carping to complain that the title of the paper by D.J. Kewley in the February 1981 issue of the ACJ (Kewley, 1981) is misleading, but the prevalence of misconceptions in language issues is so high that it warrants mentioning. The title is "A Comparison between Pascal, Fortran and PL/I". Though the introduction to the paper hints at some wider goals, it is abundantly clear from the explicit statements in the Introduction and in the body of the paper that the topic of discussion is not a discussion of the merits or efficiency of these languages, but is an examination of the efficiency (and time-efficiency specifically) of various implementations on numeric and semi-numeric programming problems.

The difference is not minor; an examination of the languages requires a very great amount of evidence and careful argument to show where "expensive" features may lie, whereas a comparison of implementations requires simply examination of a much smaller number of specific results. Accordingly, I make a plea for more accurate titles for papers; this particular example could have been "A Comparison of Pascal, Fortran and PL/I compilers".

IMPLEMENTATION DETAILS

To point up the importance of distinguishing between a language and any implementation of it, it may be interesting to cite my experience at the UCDS San Diego Conference on Pascal Extensions a few years ago. The participants at the conference were presented with a gigantic "wish-list" of extensions when they arrived, and the proposers were clearly hoping to transform Pascal into something quite different.

My analysis of the situation was that of this long wish-list, probably about 10 percent addressed real language problems, and about another 10 per cent addressed issues which might be useful but one didn't really know how to resolve the differences of opinion. The rest (thus about 80 percent) were derived from misunderstandings of the difference between a language and an implementation. Some people were asking for language features to be added to Pascal to permit them to do what they already had permission to do!

To cut a long story short, the progress of that Conference then became a mass educational exercise in training participants to recognize the difference between a language and an implementation. The point was driven home forcibly in the later sessions when a speaker on some pet topic would face a questioner who would simply ask, "But isn't that an implementation detail?" and sit down. Usually followed by collapse of argument and speaker.

CONFIRMATION

However, another point of this communication is to confirm that the findings of Kewley are rather indicative of what others will find when they compare implementations of Pascal, Fortran and PL/I on numeric problems. Pascal will usually execute slower than Fortran, usually within a factor of 2 or 3 and often much less.

Why are Pascal implementations of this quality? The question is interesting, because there is very little in Fortran (other than FORMAT and its I/O) which is not also in Pascal, and there is no a priori reason to think that implementations of comparable quality should be any different. Indeed Pascal might have an edge with its whole-array or part-array assignments.

The reason, I think, is bound up in several issues:

• Most of the Pascal implementations are not fully polished products and do little optimizing. This is a consequence of their amateur origin and the little time that has elapsed since manufacturers began to fiddle with them.

• Most Pascal compiler writers are not unduly concerned about array speed or numeric processing. The issues that worry them are component access, dereferencing pointers, packing of data structures, and procedure calls.

• A greater concern for correctness can inhibit some optimizations which are only partly equivalent to the effect defined in the language.

• Some of the demand for Pascal is derived from its compilation speed and from its (usually) greater detection capability of errors. Emphasis on these features often leads to losses in execution speed.

Of the three compilers tested by Kewley (IBM, Univac and CDC), I would assess only the CDC one as being of full professional quality at this date, though the others are clearly improving. (Note that the IBM compiler used by Kewley is not the supported IBM product — Pascal/VS). However, in saying this, it is important to note that the CDC compiler runs on a machine with ridiculously little memory given its processor speed, and the compiler optimization effort is rather more directed to producing compact code and compact data representations.

RUN-TIME CHECKING

One of the areas Kewley notes as a probable cause of poor Pascal compiler performance is the existence of run-time checking of properties of the program, as compared to the typical Fortran compiler which does few (if any) such checks. There are two points to make in this regard. Firstly, it has been shown that a little extra work in a compiler can optimize out practically all the run-time checks (Welsh, 1978), so that only the really crucial ones remain in the generated code. Probably the best example of a professional-quality Pascal compiler that does this is the ICL 1900 compiler. Extensive tests on this compiler have shown that the speed penalty for leaving checks in can be reduced to about 5 percent on average, and sometimes down close to nought percent. I wouldn't even contemplate turning checking off and making programs untrustworthy for so paltry a difference in speed.

Secondly, and it is hard to repeat it too often, correctness is a jewel beyond price. In a development environment it is quite easy to see that a run saved is equivalent to an increase in speed, but it is equally true to say that correctness has an important effect on production environ-
ARRAY ACCESS

I think it is true to say that Pascal compilers do not generally pay as much attention to array access (indexing) as they might, probably because numeric processing has not been hitherto a main application area for the language. This manifests itself in three losses of opportunity that many optimizing Fortran compilers pick up:

- Identical index expressions, or similar index expressions, in a single assignment statement may be optimised into one index calculation with slight modifications in use. Most Fortran compilers would do this, but perhaps not yet a majority of Pascal compilers.
  
  
  Pascal  
  \[ a[i] := b[i - 1] + b[i + 1] \]

- Replacement of index re-calculation within loops by some form of calculation from the previous value (a recurrence relation) is often done in optimizing mode by Fortran compilers. Although it is even more likely to pay off in Pascal compilers and just as easy (or hard) to detect that it is possible, it appears far less frequently in them.
  
  Example: \( DO 99 I = 1,N \)  
  
  Fortran  
  \[ C(I,J) = D(I)*2 \]  
  
  Pascal  
  \[ c[i,j] := d[i]*2; \]

- In computers with an array processing feature, the possibility of using it may not be detected. Such features exist in many machines, ranging from whole array processors such as ICL's, vector processing in the top range CDC machines and Burroughs' 6700/7700 range, and even machines with fast-loop facilities.
  
  Example:  
  
  For i := 1 to n do begin  
  c[i,i] := d[i]*2;  
  end;

These examples above are still relevant to this optimization. There is one area where Pascal compilers offer a feature that Fortran can only do by optimization, and that is in whole-array transfers. Pascal permits simple assignment of arrays or sub-arrays, e.g.:

\[ a := b; \]  

both \( a \) and \( b \) same size  

\[ c[i] := d; \]  

c is two-dimensional

which is normally implemented by a block transfer if one exists, or a fast-coded loop otherwise. Of course one would not expect programs simply transliterated from Fortran to Pascal to contain any such constructs, though Pascal programmers would naturally use them.

AREAS TO SEARCH FOR INEFFICIENT IMPLEMENTATIONS

Apart from the two cases mentioned above, I cannot think of other areas of Pascal which are commonly inefficient in most implementations. There are areas in which some compilers are deficient, and the cases that spring to mind are:

- sets, which are sometimes very minimally supported;
- packing, especially of subranges, which is only at an obvious level;
- case-statements, which are often restricted to jump-table implementations.

None of these would be likely to appear in a Fortran program simply transliterated into Pascal.

It may be of interest to remark that one of the major areas of inefficiency in Fortran – the overheads associated with \textsc{format} – are usually much better handled in Pascal.

The series of Pascal “validation reports” now appearing in Pascal News (Mickel, 1979 and Shaw, 1980) should be of interest to those examining the relative merit of Pascal implementations. The forthcoming revision of the Validation Suite will considerably extend the number of quality-measuring tests.

SUMMARY

The quality of Pascal compilers is indeed capable of improvement, especially in handling numeric programs. If we couple this with the observation about distinguishing between languages and implementations with which I opened this short communication, it becomes clear that the appropriate response to this situation is not to conclude that Pascal is inferior to Fortran, nor that in some cases users must lapse into Fortran to gain speed.

Rather the best response is to write a swift and nasty letter to the supplier of the compiler, asking as directly as possible why their compiler is not up to the standards expected of a professional product, and what they are doing about it. Other responses then become temporary palliatives. If users do not complain, then they have only themselves to blame for the quality of the products they receive.

REFERENCES


Pascal validation reports (Burroughs B6700, OMSI-1, P4), \textit{Pascal News, No. 16, 1979.}

Pascal News validation reports, (VAX, Apple II, Pascal-VU, CDC release 3, TI-Pascal), \textit{Pascal News, No. 17, 1980 and (3 compilers for IBM machines, B6700, VAX 11/780, IBM Stony Brook, Univac 1100 Wisconsin), Pascal News, No. 19, 1980.}


REPLY TO SALE

D.J. Kewley*

*The author is with the Defence Research Centre, Salisbury, SA, 5078. Manuscript received 20 April 1981.

Before entering into detailed comments about Professor Sale’s remarks, I emphasize that my paper was written from the viewpoint of a user in a scientific and engineering environment where computer usage (CPU time) is accountable. Consequently a user can be faced with a choice of software languages that is dependent upon the final cost for CPU execution time used and not the differences between a language and its implementation. If a language implementation does not do justice to the theoretical superiority of the language for an application then quite rightly it is a matter for concern. However, the user is faced with whatever is available now and when time-consuming programs (e.g. for signal processing) are to be run over a considerable period the final cost becomes overriding.

I was pleased to find Professor Sale confirming that my results for speeds of Pascal implementations relative to Fortran are typical for numeric problems. I would be very interested to know of any further published references.
of which he is aware, particularly with regard to PL/1. The
differences between Fortran and Pascal are further dis­
cussed in detail under the assumption that only Fortran-like
constructs are compared and that Pascal has inherent
features that will sometimes improve execution speed. This
would be expected in practical implementation and my
results show that particularly Format and Edit are inferior
to Write while Pascal recursion is at least as efficient as the
PL/1 implementation. Detailed comparisons between
Pascal and PL/1 of non-Fortran features, such as simple
assignment of arrays, were not presented, although simple
to do, because Fortran with its large number of libraries is
the major scientific language at the present time. The new
USA Department of Defence language ADA, based upon
Pascal, will be required to provide fast execution speeds for
real time applications and tests of the type presented will
be needed.

The author shares Professor Sale's view that 'correct­
ness is a jewel beyond price' and believes that runtime
checks which have only a small penalty in speed are needed.

In summary,

1. I did not conclude that Pascal is inferior to Fortran as
   suggested by Professor Sale. I did conclude that exe­
cution speed was inferior on average for the examples
   presented;

2. I did summarise speed comparisons of their imple­
   mentsations which left the user to decide on his
   choice of programming language. For the question:
   "Does the speed cost outweigh the advantages of Pas­
cal over Fortran and PL/1 on the IBM 3033?" I sug­
gested a compromise solution be taken while waiting
for a superior Pascal compiler. Although Professor
Sale agrees that Fortran usually executes faster than
Pascal he seems to have overlooked the user's respon­
sibility for taking advantage of the faster speed to re­
duce costs;

3. Although the Pascal compiler used was the product of
   a small Australian team at the AAEC, I believe it
   compares favourably with products supplied by the
   world's largest computer manufacturer and inventor
   of Fortran and PL/1.
BOOK REVIEWS


This is an excellent and well-written book that covers all issues relevant to distributed systems. Both administrative as well as technical issues in the structure, design and management of distributed systems are described. It is intended as a general survey of current status of distributed systems. Hence it is likely to appeal to readers seeking a clear and complete introduction to the issues of distributed system development, rather than to the specialist or researcher seeking detailed elaborations of the details of some distributed system components. For this reason it is intended more for the needs of a practitioner first contemplating a distributed system rather than the academic, although it is a useful text for an introductory course on distributed systems. There are no exercises in the text for teaching purposes.

The book commences by developing a terminology which is consistently used in later chapters. It follows that with an overview of the alternative strategies for distributing systems. The general approach taken in this overview is that strategies can be developed for distributing some components of a system whereas others can remain centralised. Subsequent chapters then describe the major components that make up the distributed system namely, hardware, software, the human interface, database and the communication subsystem. Considerable attention is given to alternatives of data distribution emphasising the replicated and partitioned approaches and considering the distribution of data directories and their relationship to the database.

There are a number of chapters on design but none develop a comprehensive framework for choosing a distribution strategy from the large number that are available. Methods of establishing requirements are described and choices open to designers are certainly outlined but the recommendations as to how to make a choice are usually broad and suggest further references. This particularly applies to performance evaluation, which is not treated in the text, although reference is made to some performance models.

Following these broad issues, the book concentrates on the particular technical problems that distinguish distributed from centralised systems. This includes an excellent information overview of alternate protocol strategies, which are used to synchronise updates of both partitioned and replicated databases. Extensions of protocols for deadlock avoidance, security and reliability in distributed systems are then described.

The book concludes with a number of case studies and an extensive bibliography. Overall a highly recommended book especially for those seeking a complete and concise introduction to the current status of distributed systems.

J.T. Hawryszkiewycz, Canberra CAE


This is the latest in a well-known series that began in 1960. Each volume has tended to follow the format of one or two articles on theoretical hardware implementation, new developments in software engineering, theory of algorithms and the sociology of computing with minor variations. This volume is no exception with articles on Database Computers, The Structure of Parallel Algorithms, Clustering Methodology in Exploratory Data Analysis, Numerical Software and Computing as Social Action.

In 'The Structure of Parallel Algorithms', H.T. Kung has brought together in a consistent study current knowledge on the design of parallel algorithms for parallel computers. Unlike the preceding article this is an original research contribution that not only explores the rich variety of approaches for both synchronous and asynchronous parallel computers but suggest methodologies for the design of efficient algorithms. However, it may be some time before the theory helps the practice since parallel computers are not very common as yet.

In 'Clustering Methodologies in Exploratory Data Analysis' we return to the tutorial approach based upon work done by the authors Richard Dubes and A.K. Jain and others. Clustering methodology is about a set of techniques that can be used as an aid to classifying data. As the authors remark there is a bewildering array of algorithms available and any one of them has an ability to find clusters whether or not clusters are inherent in the data. The authors introduce and discuss many different methods, providing in each case the mathematical basis for the methods. However, while explaining the method that minimizes a square-error they have omitted the method that maximizes a square-error in mixture resolving. This latter seeks to find the hyperplane that will divide a mixture of two multinomial populations into their separate (multinomial) populations by maximizing the square of the distance of each member from the hyperplane in an attempt to cause a clustering on either side of the plane. The plane happens to be the eigenvector corresponding to the maximum eigenvalue of the matrix describing the original generation of the mixed population. Since the matrix of the data may turn out to be ill-conditioned we have not only this problem to overcome but also whether it affects the validity of the clustering.

C.W. Gear in his article 'Numerical Software: Science or Alchemy' presents one man's thoughts on an area that in recent years has tended to be of less concern to computing people than in the past. He admits that while error analysis is the core reason for numerical analysis, numerical software shows more alchemy in its development than science. While this article is a useful summary for the budding worker in this field it would hardly seem to contain much that is not already familiar to the expert.

The final article by Rob Kling and Walt Scacci on 'Computing as Social Action: The Dynamics of Computer Complex Organizations' is like so many sociological studies — full of descriptions of problems but light on solutions. Many would, of course, argue that unless we can understand the problems we cannot solve them. However the choice of classification remains one of opinion and may not be the most appropriate description for the solutions being sought. This article also seems to make the assumption that all applications of computers are fundamentally similar, thus overlooking the significant sociological differences that arise as between 'big' computer, mini-computer and micro-computer applications. The authors conclude by implication that (all) 'computerization helps increase the concentration of selected consumer industries', but many would dispute this assertion as being too general.

The article is, never-the-less, a useful introduction to the novice into a developing field and provides much in the way of possible rationales for the introduction of large computers and systems into large organizations. A careful reading may help the implementer avoid some of the pitfalls of implementation in the wider environment of users who are unsympathetic towards computer technology.

The articles of the book are, as usual, well annotated with references and there is an author index of references made within articles as well as a subject index. The publishers have also thoughtfully provided a complete list of articles in earlier volumes of Advances in Computers. No doubt many will find it useful as the earlier volumes as a reference aid and a help to thought about the subject matter covered.

D.L. Overhew, Canberra CAE

The first paper is an excellent summary, albeit brief, of contemporary models used in office system design, with particular emphasis on the structure and application of procedural models. Three papers then discuss specific experimental designs (e.g. KAYAK, VIOLET projects) as practical models for the integrated office systems architecture. The third and fourth themes concern the office perspective, applications and office information modeling. Individual papers differed widely in subject matter, covering methodology, the organisational context of office systems, a conceptual approach to office automation, the use of computerised PABX as a switch in an office network and mathematical techniques for stripping office automation flow offices. A major criticism of Richer's work on the design of an office network is that if a single communication network handles both voice and data, then the network should be optimised for voice. Of the remaining papers, the contributions on networking and person/machine interaction were felt to be of particular merit in terms of the richness of the ideas presented and high quality of the practical implementations reported.

In general terms, the main impact of this unique collection of papers and working party reports is to drive home to the reader the considerable complexity of integrated office systems. Future progress in office automation is more likely to come from careful analysis of user information requirements and communication needs than from the technology driven solutions more usually offered in the seventies. This theme was very evident in Tapscott's paper on "Towards a Methodology for Office Communication Systems Research", in which Burns (1978) is quoted: "... The associated risk is that these systems (office of the future) tamper with the most sophisticated process I know of ...". The office, although we take it for granted, it is the product of 200 years of development and refinement, and changing it will require our best systems planning skills.

This underlying message that the automated office will probably emerge as one of the more sophisticated and ambitious examples of future communication engineering projects, thereby exposing the full challenge of office systems design to our professional skills, is the highlight of the book's impact. My only regret is that this theme was not developed further. Management information system needs requiring textual databases. However, the book is strongly recommended to all EDPS professionals interested in office systems design and/or advanced communications technology as a scholarly contribution worthy of inclusion in any "state of the art" library or personal collection.

B. J. Garner, Deakin University


The importance of adequate controls in computer-based accounting systems is highlighted in this extremely readable book aimed at financial managers and business students. The author delivers a lucid relationship between management accounting, auditors and system designers, and their relative responsibilities for ensuring that appropriate controls are included in computer systems.

In his initial examination of a computer-based accounting system in Chapter 2 is somewhat naive (which Edwards admits), but it does emphasize the problems if a system is designed to produce only final reports from source data. A general definition and structure of controls, and a brief overview of auditing objectives and techniques, are presented in this chapter.

Processing Controls includes a cursory examination of management, alternative organisation structures, acquisition of computer resources and personnel, performance standards, operations manual and financial planning and control. Many of the financial management and accounting controls are "... the same for both manual and computer systems" (p. 108), yet little attention is given to distinguishing which controls need to be restated, changed or deleted in moving from a manual to a computerised environment. In the examination of the control procedures and techniques for the finance, expenditure, production and revenue cycles, it is not always obvious which controls and procedures are relevant for computer-based systems as compared to manual systems. As a result of attempting to relate controls to the context of financial management systems, it is not until Chapter 6, "Processing Controls" that transaction processing within a computer system is examined in detail.

The important chapter on Processing Controls includes a comparison of manual/bookkeeping machine systems with computerised systems, and a comprehensive discussion of the specific controls that may be applied to accounting systems. These include recognition controls (source documents, input preparation), acceptance controls (data conversion, entry and acceptance), processing controls, database controls, interface controls and output controls. An excellent feature is the classification of controls into risk categories.

The final section deals with evaluating and implementing controls, and a brief overview of auditing objectives and techniques. A useful case study of an accounts payable system illustrates the control elements needed to satisfy accounting objectives, and highlights several of the processing controls examined previously. It is curious that the authorised invoice (for payment) is the first transaction input to the computerised system, presumably the matching of purchase order, goods received report and invoice being undertaken manually. Greater control over transaction processing in such a system could be obtained by computerising the purchase order, goods received report, invoice matching procedure.

It is disappointing that the unique feature claimed for the book - "... that it integrates the use of computers into the accounting process, rather than dealing with it as a separate subject ..." (preface) was not achieved. Unfortunately, the separate discussion on general management and accounting controls which preceded the important chapter on processing controls thwarts such a claim to integration. Despite these blemishes, the financial manager, system designer and business student will find the book an invaluable insight into the objectives of controls in computer-based accounting systems.

Stewart A. Liech, University of Tasmania


There is always a use for a set of carefully chosen articles in undergraduate and graduate courses, and Riley: Management Information Systems is no exception. This book of readings provides an up-to-date selection of articles containing ideas and insights - "Introducing MIS", "Planning the MIS", "Designing the MIS", and "Implementing and Evaluating the MIS". It would be suitable as a source of supplementary readings in broadly-based MIS courses. Each of the four divisions is prepared with an introduction, which provides a useful synthesis for the articles which follow.

Part 1, "Introducing MIS", includes articles which discuss systems theory, the philosophy behind the systems approach, the evolution of MIS, the utility of minicomputers and distributed systems and an introduction to some of the objectives of DBMS. Included in the selection is Ackoff's well-known article of 1971, "Towards A System of Systems Concepts" which examines the relationship between a system and its parts, and concludes with the not unexpected remark that "Systems thinking, if anything, should be carried out systematically".

Part 2 outlines the planning process with special emphasis on strategic planning. The six articles in this division cover a wide range of topics, including the impact of computer-based MIS on organisation structures, the importance of user involvement in systems development, and MIS planning in distributed data processing systems. Each of the articles is accompanied by an introduction which precedes several articles on "Designing the MIS" in Part 3. The broad coverage in this division includes the organisational content of MIS and user-centred software designs, the evolution of design approaches, database and distributed processing systems.
In Part 4, emphasis is placed on "... the importance of successful implementation and careful evaluation of the MIS". Alternative solutions, the need for management support, the application of cost-benefit analysis and evaluation by systems audit are all considered in the wide range of articles. This collection is well-organised and is a valuable reference source for use in industry-focused systems courses. As would be expected, a reading of a book of readings, a rigorous analysis of planning, designing and implementing MIS is not provided. However, the book would be of interest to managers of organisations who require a general insight into MIS to students who require current supplementary articles for use with a major text.

Stewart A. Leech, Department of Accounting and Finance, University of Tasmania


This is a large book (printed on American quarto paper) which consists mostly of diagrams, tables and other manufacturer-supplied information about microprocessors, with special attention to Intel 8086, Zilog Z8000 and AMD2901. The amount of explanation for any subject, be it 8086 instruction set or memory and I/O interfacing, is far from extensive, and the reader is more or less expected to wade on his own through the manual-style descriptions. Although not at all useful for introductory purposes or as a textbook, the book brings under one cover a very large amount of useful material, and a person with a good background in microprocessors will find it quite informative. At the end of the book are a list of microprocessor models and a section on board microcomputer systems, as well as a glossary of terms, quite complete considering the rapid development of the field, but the bibliography is far too short, though some chapters provide a few additional references. The production quality is high, legibility of the reproduced material is undoubtedly a good one, and meets a market requirement not currently satisfied. However, it seems to me that the publisher (very active in the computing field and likely to start other initiatives too) might have over-reached itself in launching six separate series in the subject, and while the material in the three volumes provided to the journal seems to be quite reasonable, I do not quite see that, whatever rapid progress EDP might make, there could be quite enough significant advances every year to fill six books with good original material. Indeed, even with the current three books there is already some overlap, e.g., the paper on computer security in the first book obviously overlaps the subject of the Computer Security Management Series. I suspect overlapping and repetition will become increasingly pronounced as the series go on year after year. We shall see.

As each book consists of contributions from a number of authors on various aspects of its subject, quality tends to be uneven. Generally I find papers discussing fairly concrete subjects the most informative, e.g., chapter 2 of the first book on costing the acquisition of small computers, and that of the third book on programming aptitude tests. A few papers, however, show the usual tendency of management studies towards waffle, and one, chapter 12 titled "Future Computers: Impact on Data Processing Management" in the first book, can only be called buzz-word-filled drive! Typical of the repetitive and uninformative passages are sentences like "... move should be towards smart people amplifier devices." This collection of future computers or smart DP systems includes wearable or carriable smart management machines, smart programmer machines, smart doctor machines, smart auditor machines, smart steel, and quite a few more elsewhere in the paper. Fortunately, the chapters are not all like that.

I found the book on database management a little disappointing. I can see that the authors are deliberately pitching the material at a not-too-technical and not-too-theoretical level. Unfortunately, as this is a complex subject, if the theory and technical details are avoided then one is left with very little to say. Perhaps a change of title to "Advances in Database Administration" would reflect the contents more accurately.

Despite my misgivings, I believe the book will be of use to people involved in EDP management. I shall be interested to see how things shape up in the next few years.

C.K. Yuen


This torrent of books on microprocessors is something of a nightmare for the Editor, for reviewers are a bit hard to find on this subject — perhaps people in the field are all too busy programming and interfacing their micros! So instead of sending these books to nine different prospective reviewers, I will do them all myself.

These nine are originally all meant to be low cost products for everyday micro users, though their Australian prices are far higher than US prices (at least for the Z80 books). As do-it-yourself-type books go these are quite comprehensive, though because they have not been used in actual teaching there are sometimes obvious lapses of detail, e.g., nowhere does the Z80 book explain how to switch from one set of general registers to the other; indeed the alternative "A to 'L" were mentioned a couple of times and then virtually forgotten. However, where the Sybex books win out is that there are books on just about any subject you can think of, whereas other micro books tend to be more general and hence take longer to get to the particular machine one wants to know about. Educationally of course that is right, but micro users may well feel the Sybex approach is preferable. Take your choice!

C.K. Yuen


The idea of having an annual series of books on EDP management is clearly a good one, but the market requirement is currently not satisfied. However, it seems to me that the publisher (very active in the computing field and likely to start other initiatives too) might have over-reached itself in launching six separate series in the subject. While the material in the three volumes provided to the journal seems to be quite reasonable, I do not quite see that, whatever rapid progress EDP might make, there could be quite enough significant advances every year to fill six books with good original material. Indeed, even with the current three books there is already some overlap, e.g., the paper on computer security in the first book obviously overlaps the subject of the Computer Security Management Series. I suspect overlapping and repetition will become increasingly pronounced as the series go on year after year. We shall see.

As each book consists of contributions from a number of authors on various aspects of its subject, quality tends to be uneven. Generally I find papers discussing fairly concrete subjects the most informative, e.g., chapter 2 of the first book on costing the acquisition of small computers, and that of the third book on programming aptitude tests. A few papers, however, show the usual tendency of management studies towards waffle, and one, chapter 12 titled "Future Computers: Impact on Data Processing Management" in the first book, can only be called buzz-word-filled drive! Typical of the repetitive and uninformative passages are sentences like "... move should be towards smart people amplifier devices." This category of future computers or smart DP systems includes wearable or carriable smart management machines, smart programmer machines, smart doctor machines, smart auditor machines, smart steel, and quite a few more elsewhere in the paper. Fortunately, the chapters are not all like that.

I found the book on database management a little disappointing. I can see that the authors are deliberately pitching the material at a not-too-technical and not-too-theoretical level. Unfortunately, as this is a complex subject, if the theory and technical details are avoided then one is left with very little to say. Perhaps a change of title to "Advances in Database Administration" would reflect the contents more accurately.

Despite my misgivings, I believe the book will be of use to people involved in EDP management. I shall be interested to see how things shape up in the next few years.

C.K. Yuen


It has been interesting to once again read this important book in its second printing by the University of Queensland Press. Even though it was recently recently reviewed in ACE, its timeliness, its relevance (particularly to the Australian scene) and its overall results and recommendations. The book is divided, I feel, into four separate sections:

(a) Chapters 1 to 4, covering an introduction, a broad overview entitled "Towards the Information Society", along with two chapters (numbers 2 and 4) outlining "The Study Approach" and "The Technology and Forecasting".

(b) Chapter 5, a small chapter on the "Structure of the Computing Industry".

(c) Two chapters (6 and 7) on uses of the technology and the implications of its impact on technology, and perhaps the allocation of computer resources.

(d) Chapters 8 and 9 covering "Policy Proposals" following from the previous studies using the methodology outlined and "Policy Implications" expanding on the broader impact of proposals that are made.

In the foreword to the book the overall theme is set by Sir Ian Maddock, while commenting on a 1970 visit to Japan, when he states: "For as far as any outlook on the future of technology can reach — well into the next century, the ability to gather, record, organise, analyse and act upon information is going to be a dominant factor. What steam, steel, electricity was to the nineteenth century, information management and exploitation will be for the
next half century if not very much longer."

It should be borne in mind that the work reported in this volume was begun in August 1976, with some amendments made in early 1978. So, we are now three years past the original study, and as such it would be most worthwhile, rather than concentrating on the technology reported in the book to examine the policy recommendations to the UK Government are worth starting with:

(a) It is far more important that the UK use the new technology than that the UK should provide it.

(b) The Government should consider reorganising the Government Research Establishments (GRE) resources it has available into a single Institute for Information Technology. In broad view this reviewer cannot support proposition (a), nor does he feel that the evidence presented in the book leads to such a wide and dogmatic statement, either for the UK or for Australia, or indeed, any "developed" country. The second proposition is somewhat easier to evaluate and on the surface would appear to be sensible both in the UK and Australia, providing it would a "centre of excellence". Concern could, however, arise as to the relevance of such an institution in relation to a developing industry.

Further Comments on Early Chapters

On the whole, the early chapters of the book become somewhat tedious, in that, one obtains essentially lists of expected developments or broad categories of technologies which may or may be not pertinent later in the book. There are, however, some telling statements "hidden away" in these chapters, including in Chapter 4 on software: "Because of the failure of the market mechanism in the area of programming languages, there is a strong need for government involvement in both the development and promulgation of language."

The emphasis is that of the reviewers in discussing information technology it must be borne in mind that it appears that normal private industry and investment seem to fail. In Australia this has been called the "venture capital" drought. The normal business or financier, may be willing to place capital in a company in an already existing and proven overall industry, but, never in a company which is itself participating in a new industry, unknown to the financier. This Australian malaise is obviously a UK problem as well.

Recommendations

Section (b) as I have called it should be a large and important section, detailing as it should the broad political and economic consequences of the current structure of the information technology industry. Unfortunately, this section is only one chapter of ten pages and is totally inadequate even for the UK scene, and has no real significance for Australian analysis. Section (c) is somewhat larger and gives an overview leading to section (a), the important section on proposals and likely impacts of those proposals. The policy section (chapter 8) is rather hard to translate to the Australian scene, being dedicated to the basic organisations and government groups behind the report, but, once again the emphasis is on one of applications or use of the technology. This is put succinctly in the statement where the present reviewer would appear to be a considerable risk that the UK will fail to exploit the technology as aggressively as our industrial competitors.

Policies to assist in reversing this trend in the UK are thus proposed by Barron and Curnow.

Summary

To this reviewer the Barron and Curnow report is a disappointment. At the first level the report appears defeatist - that the UK must remain a "user" rather than a developer and exporter of the new technology. The second point is one only now coming to light. The clear separation Barron and Curnow make between the technology itself and its applications required by them in their sections on the software side of the industry is starting to fail in the 1980s. In other words microelectronics is now being seen as a "tool" used to build a "product"; a product that can be sold to a "user". In other words the patch of component supplier to system developer to application developer to user is simply getting shorter. The component manufacturers are moving "up market" to systems and to applications, (e.g. word processing, electronic mail and funds transfer, and so on). Meanwhile the systems builders, e.g. Digital Equipment Corporation of the USA, are building their own IC competence. The world of microelectronics has become the "lathe" of the twentieth and twenty-first centuries. A company sells products produced by its "lathe" and herein lies the fundamental difference this reviewer has with Barron and Curnow. The clear separations of roles made by the authors are fading in the 1980s. This should have been obvious even in the mid-1970s. The suggestion that a country's future could be based on "use" of the technology alone cannot be agreed with. It is a short step to the "technological colonialism" of the future.

W. I. Caelli,
Electronic Research Associates,
Chairman, ACS National Committee on Hardware Technology and Hardware Industry


This book is the proceedings of a workshop held in Berlin in 1979. Like all such publications compiled in this manner, it suffers from a lack of coherence with considerable variations in the quality of individual papers. As many of the original presentations were made in German, there is an added complication that these had to be translated into English before publication and in some cases this is very obvious.

Following on from an excellent introductory paper on "Program Adaptation and Program Transformation" by James Boyle of the Argonne National Laboratory, there are sections with papers describing accounts of actual conversions, use of tools, testing techniques, machine supported library organisation, user oriented aspects and documentation. Although the titles of some of the papers appear quite interesting, the contents often turn out to be not quite so fascinating, consisting merely of a rather dreary account of how program "x" was moved and adapted to machine "y". The last section which is probably the most useful in the entire book is a very comprehensive 10-page bibliography of methods and tools for software adaptation and maintenance with a corresponding Kwc Index.

In summary, I cannot recommend this book as one likely to be of general interest. It could be a useful reference for persons actually involved in the task of moving and adapting software particularly if SPSS or the NAG or IMSL subroutine libraries are involved.

Peter C. Poole,
University of Melbourne


This volume reports the proceedings of the First International Conference on Entity-Relationship (E-R) approach held on December 10-12, 1979. Like all such publications compiled in this manner, it suffers from one of applications or use of the technology. This is put succinctly in the statement where the present reviewer would appear to be a considerable risk that the UK will fail to exploit the technology as aggressively as our industrial competitors.

Policies to assist in reversing this trend in the UK are thus proposed by Barron and Curnow.

Summary

To this reviewer the Barron and Curnow report is a disappointment. At the first level the report appears defeatist - that the UK must remain a "user" rather than a developer and exporter of the new technology. The second point is one only now coming to light. The clear separation Barron and Curnow make between the technology itself and its applications required by them in their sections on the software side of the industry is starting to fail in the 1980s. In other words microelectronics is now being seen as a "tool" used to build a "product"; a product that can be sold to a "user". In other words the patch of component supplier to system developer to application developer to user is simply getting shorter. The component manufacturers are moving "up market" to systems and to applications, (e.g. word processing, electronic mail and funds transfer, and so on). Meanwhile the systems builders, e.g. Digital Equipment Corporation of the USA, are building their own IC competence. The world of microelectronics has become the "lathe" of the twentieth and twenty-first centuries. A company sells products produced by its "lathe" and herein lies the fundamental difference this reviewer has with Barron and Curnow. The clear separations of roles made by the authors are fading in the 1980s. This should have been obvious even in the mid-1970s. The suggestion that a country's future could be based on "use" of the technology alone cannot be agreed with. It is a short step to the "technological colonialism" of the future.

W. I. Caelli,
The broth is thickened further when it is realised that few of the proposed conceptual modelling methodologies is matched by an efficient and effective commercially available DBMS. It appears that much time will pass and much experimentation be completed before we find a conceptual modelling method which is:

- suited to a range of users with differing views of reality and differing application complexity,
- readily subjected to evolutionary changes,
- secure from unauthorised and semantically inconsistent operations and which
- maps readily to an efficient and flexible implementation level structure.

Today's application systems programmers must define nearly all the desired semantics in tailor made application programs rather than as part of the data, with potentially disastrous results. The availability of DBMS which allow explicit declaration of all semantics will be an important forward step in this field. In the meantime we will have to make do with TOTAL, IMS, DMS II, IDMS, System R, etc.

This book is suitable for the advanced practitioner who wishes to know what facilities will be available in Commercial DBMS in 10 years time. In addition, the book is essential reading for all tertiary teachers, researchers and implementors of DBMS software systems, all of whom will find a wealth of ideas about facilities that could be made available to users in a DBMS.

For these two groups of people, the book is worth every cent of its extra-ordinary price.

A. Y. Montgomery,  
Monash University


This book is subtitled "The Complete Guide to the New Technology and its Impact on Society", and this is exactly what it is. There are 42 reprinted articles presented in three sections with a total of nine chapters. To give some idea of the completeness of the coverage let me list the chapter headings:

1. The New Technology
2. The Microelectronics Industry
3. Applications of the New Technology
4. The Impact In Industry
5. The Revolution in the Office
6. The Consequences for Employment
7. Industrial Relations Implications
8. The Social Impact of Computers
9. The Information Society

Each article is written in a clear, direct style (intended for the layman but pulling no punches) by a recognised authority in the relevant field. Each chapter is accompanied by a guide to further reading. The book terminates with an excellent Glossary of Terms. The book is very much up to date with the oldest article originally appearing in 1975. Articles are drawn from Datamation, Scientific American, Business Week and other sources with most date lines in 1978.

Every practitioner really interested in being able to discuss this important topic knowledgeably should buy this book. In addition, every teacher of courses on the Social Implications of Computers will need to have this on his shelf for reference and for a prime source of class material. Every library, whether it be school, state, council, industrial or other should have a copy of this text.

The book will appeal more to the specialist than to the general practitioner. It is nevertheless a welcome contribution to an area which has been sadly neglected.

A. Y. Montgomery,  
Monash University


One phenomenon that can be observed readily in the evolution of Computer Science is the manner in which it has drawn on quite diverse subject areas as these because major sources of applications interest or of necessary development techniques, absorbed what has been needed, and passed on to other applications areas with new technological requirements and different problems. Frequently this interaction has resulted in an enormous increase in interest in the original subject, and even to a revaluation of the relative importance of its different aspects.

Interactions between Computer Science and the Mathematical Sciences are close, and of continuing mutual benefit. Historically, they have provided much of the impetus that has transformed such subjects as Numerical Analysis, Formal Systems, and Complexity Theory into important and flourishing branches of (Applied) Mathematics. However, it seems likely that many of the participants have not understood the strong pragmatic element in the Computer Science side of the exchange, and as they become more deeply involved in the formal questions raised by theoretical considerations of the special subject they argue with increasing persistence that they are providing the theoretical basis for Computer Science. Unfortunately their chosen avenue has not always been taken notice. This crisis of identity could affect the reception of the book under review and it appears legitimate to ask "is it appearing in the right series (ACM Monographs)?" and it is being reviewed in the right journal? A General Theory of Optimal Algorithms is certainly a tempting title, and the authors claim to subsume most
The starting point will be familiar to readers of this Journal namely tuples \( (a, z) \) where \( a \) is a "computing device" and \( z \) an "action": one can abstract the set of all tuples \( \Theta \) called \( \Theta \)-computable if there is a code \( a \) such that the above equivalence holds. Not every set \( \Theta \) yields a reasonable computation theory. One must put in some basic functions and require closure of \( \Theta \) under certain properties.

Generalised recursion theory is many things. One aspect starts from ordinary recursion theory and moves up in types i.e. includes not only natural numbers (elements of \( \mathbb{N} \)), but also functions \( f: \mathbb{N} \to \mathbb{N} \), and computable functions \( \psi: [\mathbb{N} \to \mathbb{N}] \to \mathbb{N} \), etcetera. Another aspect looks to domains more general than \( \mathbb{N} \) such as ordinals or admissible sets. Accordingly, many of the problems of interest are on the border between set theory and recursion theory.

For those not wishing to read far into generalised recursion theory, but who would like some acquaintance with the directions taken, the opening chapter of the book, titled 'On the Choice of Correct Notions for the General Theory', is highly recommended. Here the author outlines the connections between computation theories, finite algorithmic procedures, and inductive definability.

Those who proceed further will appreciate the author's willingness to offer frequent and informal explanations of what is being attempted and of the key points tying various topics together. His ability to provide concise insightful remarks in company with the formal and rather abstract developments of the subject is a feature of the book.

A list of the remaining chapter titles gives an indication of the topics covered — General Theory: Combinatorial part; General Theory: Subcomputations; Finite Theories on One Type; Finite Theories on Two Types; Admissible Prewellorderings; Degree Structure; Computations over Two Types; Set Recursions and Higher Types. Each chapter is divided into about four sections and these sections generally start with interesting remarks on the development of the key notions and pointers to related results in the literature.

In summary, this is an excellent book addressed primarily to mathematical logicians, but with certain aspects of interest in theoretical computer science.

Elizabeth Sonenberg,
RAAF Academy,
University of Melbourne


I must admit that software engineering, like psychology, has until recently been a term which, for me, has had almost no meaning; an ill-defined collection of largely unrelated buzz-words and verbose pontifications. So, having read this book I must heartily recommend it to all, especially educators and students.

Like psychology, software engineering cannot be precisely defined. It encompasses a large collection of inter-related ideas and theories any one of which, by itself, may be of considerable interest; but taken as a whole they reflect, Gestalt like, a philosophy of design greater than its parts.

Each paper in this book addresses, largely independently, the author's personal perspective on large system design. These approaches vary from the delightful study of the problems of learning from experience given by Boehm to the tutorial-like introductions to specific techniques such as Boyle's paper on program transformation. All seem to me to be prompted in part by the fear that, without more formal techniques and a better philosophical basis, large scale projects will forever remain out of control. And hence all represent searches, from different points of view, for a theory of design.

The worth of this book lies not so much in the particular merit of the approaches (some of which are very sketchy and debatable) but rather in that it gives the reader a framework on which to build a reasoned view of his role, and hence that most desirable of all talents, a broad perspective.

R.P. Watkins,
University of Tasmania


105
Letters to the Editor

FURTHER COMMENTS ON JULIFF’S PAPER

I refer to the file updating algorithm given by Juliff in his paper "Program Control by State Transition Tables" (Vol. 12, No. 4, November 1980).

It is worth noting that the algorithm leaves room for improvement. As a matter of detail, the references to States 6 and 9 in Procedure D should be interchanged. As a matter of principle, the algorithm fails to handle changes to newly inserted records. From the accompanying description, it is clear that Juliff is aware of this omission. However, it is difficult to see any justification for it.

For an excellent presentation of an algorithm that handle this possibility, the reader is referred to the paper by Barry Dwyer (1981). Following Dijkstra (1976, p. 117), he credits the algorithm to W.H.J. Feijen, but it may well be that other programmers discovered it independently, possibly many years earlier. Can any reader add to the history of this subject?

References


J.B. Hext,
Professor of Computing,
Macquarie University

1. There are dangers inherent in leaping from 'programming-as-an-art' to 'software engineering', not least of which is the unquestioning application of techniques presently used in some engineering (and therefore venerable) discipline.

2. This particular tool demands an intimate understanding of the problem structure, sufficient to enable the whole to be parameterised. As such it is a tool, not for the beginner or the fledgling professional, but for the specialist.

3. The first two examples given were both examples of program sub-modules, with limited complexity in their control structures (acceptance of a monetary value, and validating/reformatting of a part-number). Since these problems are masterable by a trained programmer, the use of STT, with its definite advantages in maintainability and enhancement flexibility would seem to be an appropriate technique.

4. One difficulty glossed over in the article is that not all program modifications can be implemented merely by altering values within the table. For example, if the monetary value were now to contain a letter indicating the currency (S = Sterling, etc), which letter could appear first, after the minus sign, or after the last digit, then an additional "current state" (row) would be necessary, and an additional "new state depending on next character" (column). Subsequently

5. My main concern is however with the proposition that STT be used to control program-flow. This curdles the blood of pragmatists with long memories — it was this style that was once associated with the use of COBOL ALTER and batteries of Switches named after local football clubs. That bias declared, I attempt below a more reasoned criticism.

6. That an elegant, readable and maintainable program can be written using STT I don't contest. Its use is however predicated or not merely a good understanding of the particular problem, but on a mastery of the abstract formulation of the problem-type. The majority of the software industry implements applications that are somewhat behind the 'leading edge' of the technology, using staff who are likewise. STT is not suitable for people who are not specialists in the problem area, and therefore is not a suitable general tool for program-flow control.

7. Many program-types are becoming recognised, standardised, and supported by generators. Where such a development-environment is implemented, the number of programmers who must concern themselves with the niceties of program-flow control is reduced, and these specialists can reasonably be expected to master the standard types. I therefore concede the potential importance of STT as a tool of the emerging 'methods programmer'.

8. Batch-programs are no longer a serious theoretical problem, given a suitable pre-processor with a macro-language, a library of program-structure macros for typical problems, and a pseudo-code interpreter for unusual ones. My own organisation uses DELTA, a Swiss product installed widely also in Germany, and in the UK and Australia.

9. On-line programming presents a few as-yet unmastered challenges. My own approach has been to develop from the experience of structured data-handling and treat the screen as but a special case of the driving-file; the Jackson line of reasoning is easily adapted to provide an appropriate structure and hence a structure-generating macro. The limitation is that the screen is defined as a fixed data-stream (or in sophisticated environments, several data-streams relating to several possible 'masks' in each of several possible 'windows' in the physical screen). In a wide range of commercial programs the user's problem has a relatively fixed structure that is well served by this model.

10. It is only in what I would term 'genuine dialogue programs' that the limitations of the above 'structured multi-record-type driving file' become pertinent. This is the case wherever the information demanded of the screen-user is dependent on the answers provided to earlier questions. Such key-words as 'interrogative session', 'large-scale decision-tree', and 'program-determined control-flow' (as distinct from keyboard-operator-selected) come to mind. At the banal end of the spectrum the program can skip over the field.
'Maiden-Surname' in the cases of males and never-married females. More challenging are equipment-error-diagnostic utilities and patient-interviewing programs.

11. While these programs are conceptually soluble using the above 'unit-record' philosophy I would baulk at the attempt. It is in this class of programs that State Transition Tables may have much to offer. Some of the European software houses certainly seem to think so (for example, the suppliers of the DELTA package are basing their own online-program generator on STT — Zustandsuebergangstabellen), and I suspect that many second-generation application-system generators (due say 3-5 years from now) may utilise the technique for all program flow-control.

12. I note from John Southgate's letter (ACJ 13, 1 February 1981) that at least Wacher and Partners have used the technique in commercial applications. Whether the program state is defined explicitly (Juliff) or implicitly (Southgate) is important only to the extent that maintenance is necessary. Once structure-generating macros are established and proven, the readability loses much of its importance in comparison with the memory and execution efficiencies (my own 'nicely structured' Jackson-like programs are quite adequately efficient in an online environment with human operators, but would be unacceptable in many real-time applications).

13. In response to John Southgate's "... the concept of state-control is more important as a design tool than as a maintenance aid", I suggest that we are only a little short of the point at which program-flow maintenance will be usually achieved by invoking an alternative program-structure macro, and regenerating the program using the same application-specifications. Or, put another way, our preferred design language is as of tomorrow compilable.

Roger Clarke,
Brodman Software Systeme AG,
Zuerich, Switzerland

EDITOR'S NOTE:
In view of the extensive interest aroused by the article, I am pleased to announce that a paper by Professor Hext dealing with the principles of STT will be published in a forthcoming issue.

A COMPARISON BETWEEN PASCAL, FORTRAN AND PL/I: ANOTHER COMMENT

Mr Kewley sent me listings of the Pascal, Fortran and PL/I programs used in his tests, along with copies of the compiler options used.

The compiler option OPT(2) was specified for the PL/I compilations in order to optimise time. However, none of the PL/I programs had the REORDER option included.

In the IBM PL/I Optimising Compiler, the ORDER option is assumed for the outer block, and is inherited by all contained blocks unless specified otherwise with the REORDER option (IBM[1,2]). To obtain full optimisation, the REORDER and OPT(2) options must be given. Thus, even though the compiler option OPT(2) was specified, the compiler would not have been able to carry out a full optimisation on these PL/I programs. (IBM[1], McNeil and Tracz). Furthermore, the compiler option FLAG(E) was specified, suppressing the diagnostic IEL0541 indicating that full optimisation was inhibited (IBM[2]).

The kinds of optimisation suppressed when the OPT(2) and ORDER options are chosen, include strength reduction (moving invariant computations outside loops), redundant expression elimination, inter-statement optimisations involving register usage optimisation, and loop control optimisations (IBM[1]).

The lack of optimisations could have a significant effect on the execution of the FFT program, since it contains three nested loops, one of which is nested to depth three. Since that nested loop is in a procedure called from within a nested loop, the effective nesting is of depth five.

The inhibition of optimisation would have had a lesser effect on the other PL/I programs in the set.

References

R.A. Vowels,
Dept. of Computing, RMIT

CALL FOR PAPERS

SPECIAL ISSUE ON SOFTWARE ENGINEERING

The Australian Computer Journal will publish a special issue on "Software Engineering" in early 1982. Research papers, tutorial articles and industry case studies on all aspects of the subject will be welcome, and both full papers and short communications will be considered.

Prospective authors should write to:
Professor P.C. Poole, Guest Editor,
ACJ Special Issue on Software Engineering,
Department of Computer Science,
University of Melbourne,
Parkville, Victoria, 3052,
Australia.

to notify him of their intention to submit material for the issue and provide a brief summary of their intended contribution.
of Illinois, and is an advanced interactive system with important innovations in display technology, terminal design, communication hardware, system software and courseware development. PLATO systems have been established in the United States, Canada, Europe, South Africa, Japan and now in Australia. An extensive library of courses provides instruction not only in subjects such as reading, maths, social sciences and foreign languages, but also in medical, technological and commercial areas.

The arrival of PLATO at Deakin has been eagerly awaited by Colin Hook and Robin Stevens, lecturers in the School of Education, which have both had considerable experience with PLATO as students and as authors. From 1975-79 Mr Hook was at the University of Illinois. He is probably the person in Australia most experienced with the system. Robin Stevens spent several months in 1980 at Urbana-Champaign University which uses PLATO. Both lecturers were involved in the development of instructional and research programs in reading and comprehension, music education, and in evaluating the strengths and weaknesses of PLATO as an instructional medium.

Robin Stevens has a particular interest in the use of PLATO in music education. He has worked with the Gooch Synthetic Woodwind attached to a terminal, which synthesizes up to four voices at a time and is used in aural training lessons. Dr Stevens would like to combine software development and evaluation with PLATO to make teacher training in music education more effective.

Colin Hook sees PLATO as offering a valuable alternative approach to instruction in both schools and colleges, but points out that the present high cost inhibits PLATO's widespread use.

The School of Education will be co-operating with the Gordon Technical College (which will have three PLATO terminals) in Geelong in a joint program of mutual benefit to the two institutions and to Control Data Australia Pty Ltd, which markets the PLATO system worldwide. The Gordon Technical College will be using the extensive library of PLATO lessons to provide courses in basic literacy and numeracy skills for full time and part time students.

Deakin University's contribution will be research and development in three areas: the rewriting of lessons for Australian students; the adaptation of lessons to suit local curriculum requirements; and the production of new lessons and courses in specific language and mathematics skills.

NEW WORD PROCESSOR RELEASED

Toshiba (Australia) Pty Ltd has announced the release of the first Toshiba word processor to be introduced into the rapidly expanding Australian market, the Model EW-100.

Toshiba has designed the word processor as a system that provides full text preparation flexibility with a capacity of 460 pages (295,682 characters) stored on a single-side, single-density floppy disc.

This allows for considerable text preparation and document filing capacity that will save time and office space.

It also features global search and replace, which allows an incorrect word or phrase to be replaced every time it appears in the text. Sections of standardised documents can be easily modified without having to be totally retyped.

Prestored words or phrases can be included into new texts by using the glossary function.

The whole system is designed for desk-top operation but the three major components, the display terminal, the keyboard and printer, can be positioned to suit individual operator needs and comfort.

A NEW DIMENSION IN TIME MANAGEMENT SYSTEMS

A Sydney-based company, Timewise Products, has introduced a totally new concept in simplified time management systems.

Managing Director, Mr Robert Rutishauser, says that his company offers a total package which includes assistance in the identification of individual needs, system design, installation and staff training. Timewise also monitors the first few planning projects to ensure that the client is handling the system as efficiently as possible.

Timewise planning systems are already being used extensively by Commonwealth and State government departments, overseas airlines, hospitals, hotels, manufacturing and marketing companies as well as major banking groups.

Computer companies have also recognised Timewise systems as an important aspect of computer time scheduling.

Mr Rutishauser says that "Today's visual planning aids are more flexible, versatile and easily adapted to many different applications".

"Today, more people are used to recognising signs and symbols — for example, traffic signs and directions at..."
CHINA ORDERS AUSTRALIAN COMPUTERS
Australian computer manufacturer, D.D. Webster Electronics Pty Ltd, of Bayswater, Victoria, has received an order in excess of $50,000 to supply three of its Spectrum-11 minicomputer systems to the People's Republic of China.

The company's Managing Director, Mr David Webster, said that the order had been placed on behalf of the Beijing Institute of Technology, and that he believed they would be used for educational purposes.

The order follows the successful installation of a $12,000 Spectrum system at the Institute of Computing Technology, Academy of Sciences of China in Beijing (Peking) some time ago.

This system is running the MONECS (Monash University Educational Computing System) Software. It was presented as a gift by former Victorian premier, Mr Hamer, during his first trade visit to the People's Republic of China in August 1979.

It was installed the following month by David Webster and Dr Len Whitehouse, Deputy Director of the Monash University Computer Centre and author of MONECS.

Since its installation, the machine, originally a Spectrum-11A (a computer with a 32 k/b memory and storage capacity of 315 kilobytes), has been upgraded by Webster's to a Spectrum-11B, a dual floppy drive machine with 64 kilobytes of memory and storage of 630 kilobytes. The total hardware system is valued at $A12,000 and besides the Spectrum-11 includes a hole and mark-sensing card reader, a line printer and video terminal.

The three new machines included in the order for the People's Republic consist of two more Spectrum-11B machines ($8,000 each, machine only), and a Spectrum-11B/20 ($21,615) which incorporates a dual floppy disc Spectrum with a 20 Megabyte cartridge disc drive. The Spectrum-11B/20 has a 64 k/b memory.

AUSTRALIA JOINS INTERNATIONAL ELECTRONICS STANDARDS COMMISSION
Australia has officially joined the International Electro/Technical Commission (IEC), a body set up to guarantee worldwide quality control of all components used in high technology equipment.

This was announced recently by Mr A.T. Deegan, President of the newly formed Australian Electronics Industry Association (AEIA).

The IEC objective is to establish a global system of quality specifications for minimum performance levels of components used in professional equipment in the telecommunications sphere, no matter where in the world the parts are produced.

When fully operative the IEC system will mean that all components manufacturers worldwide will have to meet the minimum standards of quality control in accordance with the specifications laid down by the Commission.

As well, importers of components will have to establish tight stock control and checking systems to ensure all imports comply with the specifications.

The governing body of the IEC in Switzerland will be responsible for setting the specifications for all components following input from all member nations.

In Australia the Standards Association will be the body responsible for administering the regulations.

Estimated annual cost of administering the Australian end of the IEC operation is put at $60,000. This will be borne by the Federal Government — 50 percent; AEIA — 33.3 percent; Telecom — the remaining 16.7 percent.

Mr Deegan said Australian membership of the IEC is strongly supported by member companies of the Australian Electronics Industry Association which represented the majority of Australia's electronics and components manufacturing companies.

He said the Federal Government is strongly behind the move also, as is Telecom which has stated it will specify components meeting IEC standards when the specifications are formalised.

"Electronic and telecommunications systems are vital pillars of modern society," said Mr Deegan.

The IEC system has been in the formative stages for the past five years. Worldwide introduction of the first specifications is expected within the next two years.

Nineteen countries including Denmark, Sweden, France, West Germany, Eire, Israel, Japan, Switzerland, UK, USA and USSR have participated in establishment of the system.

SPECIALIST HANDLING ADVICE AT DATA 81
Australia's largest locally owned firm of international freight forwarders and customs agents, John G. Stephenson and Co Pty Ltd will mount a special display at Sydney's Data 81, from August 25 to 27.

Company executives will be available to give advice on handling and quick transit techniques to companies importing or exporting computers and electronic equipment and spare parts.

John G. Stephenson and Co will have its FACS (facilitated air clearance system) facility on show at Data 81 as well as a mini-computer used for inventory and distribution.

The facsimile transmission system can cut by up to 50 percent the time it takes to import goods from the United States and Europe.

By using FACS, all documentation required by customs to allow entry into Australia can be completed before the aircraft leaves the United States, rather than after it arrives here.

Group general manager of John G. Stephenson and Co, Mr Peter Whitfield, says the movement of complex, delicate equipment such as computers calls for specialised handling techniques.

"We have established a reputation as being leaders in the field of rapid yet careful movement of computer and other electronic equipment both in and out of Australia," he said.
40 million telex messages expected.

"On our stand at Data 81 will be Ms Karen Guide, of our American associate W.T.C., who is expert in the field of moving computers and other specialist equipment," he said.

"Ms Guide and our own experts will be available to give advice on any aspect of importing this kind of equipment.

"People needing computers and computer parts, need them quickly.

"Rapid transit means a reduction in inventory costs and obviously reduces lead times for installation.

"Air freight is also demonstrably smoother than surface transport so there is much less chance of precision equipment being damaged. It also means that less packaging material is required thus further reducing costs," Mr Whитьfield said.

PHONE CHARGES UP, BUT MANY COMPANIES CAN STILL SAVE

Despite the just announced increase in telephone charges, many companies could save significant amounts by rationalising their operations, according to a telecommunications expert.

Dr Glen Hickey, a senior consultant with PACTEL, the computer and telecommunications arm of PA Australia, says few companies have any idea of the costs of their telecommunications services, nor how to go about rationalising them.

This year, he says, most of the 5,500,000,000 phone calls in Australia will be business calls, as will most of the 40 million telex messages expected.

"Until recently, communication was a simple problem of choosing the cheapest method: telephone, telegram, telex, or letter, each having cost advantages," Dr Hickey says.

"Today, however, the technological advances are multiplying the range of options available, and in doing so are laying traps for the unwary. What managers previously treated as a matter of routine is a complex task requiring specialised support if it is to be done in the most cost effective way.

Dr Hickey cited saving by two companies—one in finance and the other in the food business. The rapidly expanding national finance company rationalised its system, with the help of PACTEL, and saved $100,000 from an annual expenditure of $2 million.

The food company also saved significantly—$40,000 from an annual expenditure of $5,000.

Dr Hickey says telecommunications have become more computerised in recent years, and integration of all methods—whether by computer data lines, word processing and transmission, or telephone—is essential for control and cost-effectiveness.

3M AUSTRALIA RELEASES REPOSITIONABLE ADHESIVE NOTE PADS

3M Australia, maker of Scotch brand adhesive tapes, has released an adhesive product for offices, which in many cases eliminates the need for paper clips and staples.

The new product—Post-it Note Pads—uses a repositionable adhesive which sticks to paper and virtually all other surfaces and can be removed at any time without damaging the surface beneath.

Product Manager of 3M's Office Products Division, Donna Warren, described "Post-it" Note Pads as "a giant communications breakthrough".

3M expects "Post-it" Note Pads to quickly penetrate the estimated 400,000 strong office market with data processing divisions among the major users.

Mrs Warren said that although "Post-it" Note Pads look like ordinary yellow paper, a narrow band of adhesive on the back allows each sheet to be attached to another surface without staples, paper clips or tape.

They are ideal for attaching notes to correspondence, flagging pages in books and magazines, conveying telephone messages or attaching distribution lists, said Mrs Warren.

She expected uses of the product would expand, particularly as self-reminder notes. "Post-it" notes would become commonplace on telephones, pen-holders, and typewriters as an eye-catching reminder note which cannot get lost in the shuffle of paperwork.

"Post-it" Note Pads are available, in three sizes: 38.1mm x 50.8mm (1½” x 2”), 76mm x 2"

Mrs Warren said a white tape for paragraph blockout work, or covering changes in material to be photocopied, completed the "Post-it" Note Pad range.

The tape, 25mm x 17.7m does not show on a photocopy and may be removed at any time without damaging the original. "Post-it" Note Tape would reduce retyping and free secretaries for more productive work, Mrs Warren said.

LINKING CUSTOMER NEEDS WITH TECHNOLOGY

Sanyo Australia has launched a range of new products to major dealers from all over Australia.

Clive Reece, Creative Director of Eyeline, one of Australia's leading audio visual companies and producer of the Sanyo presentation, says that "Quality is the key to effective AV presentations".

Both Reece and co-director Ross Wall agree that clients experienced in AV presentations are demanding a much more effective presentation, both in terms of results and impact.

Since the company commenced operations late last year, Eyeline has produced dealer presentations for major clients including Acmil, Leyland, Toyota and now Sanyo Australia.

Clive Reece spent ten years in London, as producer and director for J.B. Presentations. That company's clients were heavily marketing orientated and included Pepsi-Cola, Olivetti, Colgate Palmolive, Rank Xerox, Coty, Beechams and L'Oreal.

Ross Wall joined Eyeline from the presentations arm of the Grundy Organisation where he was involved in marketing presentations for such clients as BMW, NCR, Coca-Cola and Datsun.

The end result is, the company claims, one of the most experienced teams in the field of presentations in Australia.

Eyeline says that audio visuals have, quite correctly, been seen as a communication medium for audiences, using large images for impact and the creation of a sense of size and realism.

All Reece's programmes have been for commercially orientated sales and marketing companies, where clients are seeking a specific selling message, balanced with an emotive message. His criteria is that the emotive message must be established first.
The Australian Computer Journal is an official publication of the Australian Computer Society Incorporated.


EDITORIAL COMMITTEE: Editor: C.K. Yuen, CSIRO Division of Computing Research, PO Box 1800, Canberra, ACT, 2601. Associate Editors: J.M. Bennett, T. Pearcey, P.C. Poole, A.Y. Montgomery, J. Lions.

SUBSCRIPTIONS: The annual subscription is $18.00. All subscriptions to the Journal are payable in advance and should be sent (in Australian currency) to the Australian Computer Society Inc., PO Box N26, Grosvenor Street, Sydney, 2000. A subscription form may be found below.

PRICE TO NON-MEMBERS: There are now four issues per annum. The price of individual copies of back issues still available is $2.00. Some already out of print. Issues for the current year are available at $5.00 per copy. All of these may be obtained from the National Secretariat, PO Box N26, Grosvenor Street, Sydney, 2000. No trade discounts are given, and agents should recover their own handling charges. Special rates apply to members of other Computer Societies and applications should be made to the Society concerned.

MEMBERS: The current issue of the Journal is supplied to personal members and to Corresponding Institutions. A member joining part-way through a calendar year is entitled to receive one copy of each issue of the Journal published earlier in that calendar year. Back numbers are supplied to members while supplies last, for a charge of $2.00 per copy. To ensure receipt of all issues, members should advise the Branch Honorary Secretary concerned, or the National Secretariat, promptly of any change of address.

MEMBERSHIP: Membership of the Society is via a Branch. Branches are autonomous in local matters, and may charge different membership subscriptions. Information may be obtained from the following Branch Honorary Secretaries. Canberra: PO Box 446, Canberra City, ACT, 2601. NSW: Science House, 35-43 Clarence St, Sydney, NSW, 2000. Qld: Box 1484, GPO, Brisbane, Qld, 4001. SA: Box 2423, GPO, Adelaide, SA, 5001. WA: Box F320, GPO, Perth, WA, 6001. Vic: PO Box 98, East Melbourne, Vic, 3002. Tas: PO Box 216, Sandy Bay, Tas, 7005.

CONTRIBUTIONS: All material for publication should be sent to the Editor for processing. Prospective authors may wish to consult manuscript preparation guidelines published in the May 1980 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. Descriptions of new software packages are also published to facilitate free distribution.

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 the Printer (Publicity Press). 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 2.5cm margins on all four sides. The original, plus one copy (preferably two for Papers) 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 brief Abstracts, Keyword lists and CR categories on the leading page, with authors' affiliations as a footnote. The authentic of an accepted Paper will be asked to supply a brief biographical note for publication at the end.

This Journal is Abstracted or Reviewed by the following services:

<table>
<thead>
<tr>
<th>Publisher</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>Bibliography and Subject Index of Current Computing Literature.</td>
</tr>
<tr>
<td>ACM</td>
<td>Computing Reviews.</td>
</tr>
<tr>
<td>AMS</td>
<td>Mathematical Reviews.</td>
</tr>
<tr>
<td>ENGINEERING INDEX INC.</td>
<td>Engineering Index.</td>
</tr>
<tr>
<td>INSPEC</td>
<td>Computer and Control Abstracts.</td>
</tr>
<tr>
<td>INSPEC</td>
<td>Electrical and Electronic Abstracts.</td>
</tr>
<tr>
<td>SPRINGER-VERLAG</td>
<td>Zentralblatt fur Mathematik und ihre Grenzgebiete.</td>
</tr>
</tbody>
</table>

Copyright © 1981. Australian Computer Society Inc.

Production Management: Associated Business Publications, 28 Chippen Street, Chippendale, NSW, 2008. Tel: 699-5601, 699-1154. All advertising enquiries should be referred to the above address.