INFORMATION TECHNOLOGY

1 Managing information systems in Australia and New Zealand: Requirements for the 1990s
M BROADBENT, A HANSELL, P LLOYD and CNG DAMPNEY

12 Object-oriented information systems: An introductory tutorial
B HENDERSON-SELLERS

25 Case study Cardomat/Migros an open EFT/POS system
R CLARKE

SPECIAL FEATURES

37 Book Reviews
Please record my new address as shown above. I enclose the mailing label for the last received issue.

Please enrol me as a subscriber for 1992. I enclose a cheque or bank draft for $A40.00.

Please record my new address as shown above. I attach the mailing label for the last received issue.

Send all correspondence regarding subscriptions to the Circulation Manager, PO Box 319, Darlinghurst, NSW 2010 Australia.
Managing Information Systems in Australia and New Zealand: Requirements for the 1990s

M. Broadbent*, A. Hansell**, P. Lloyd*** and C.N.G. Dampney****

*Key Centre for Strategic Management, Graduate School of Management, University of Melbourne, Carlton, Victoria, Australia
**Senior Systems Representative, IBM Australia, 168 Kent Street, Sydney, NSW, Australia
***Phoenix Contracting, Sydney, NSW, Australia
****Computing Discipline, Macquarie University, Sydney, NSW, Australia

INTRODUCTION
The management of information systems and services in industry, business and government is a key ingredient in organisational effectiveness. Information technology has the capacity to change organisational forms such that the 'new organisation' espoused by Drucker is both information and knowledge based (Drucker, 1988). The process of management itself has become information intensive (Moore, 1988).

In recent years, there has been a shift of interest in the information systems management field towards greater consideration of managerial and organisational questions, and the extent to which information systems and services are embedded in organisational processes (McFarlan, 1984; Boynton and Zmud, 1987; Broadbent and Koenig, 1988; Davenport and Short, 1990).

These interests are part of the rationale behind a series of ongoing studies of information systems management executives in Australia and New Zealand, which commenced in 1982 (see for example Dampney and others, 1984; Hansell and others, 1985; Broadbent and others, 1989). The participants in the survey are members of the Australasian Share Guide (ASG), an organisation of IBM mainframe users located mainly in Australia and New Zealand.

In February 1990, the members of the ASG were surveyed to identify their needs and priorities and to gain industry data concerning their current systems development processes. This paper reports the findings of the '1990 Information Systems Management Survey' and will make some comparisons with the 1987-88 survey (Broadbent and others, 1989). For a more retrospective review and analysis of trends, see Hansell, 1990.

SURVEY FORMAT
The study was undertaken in February 1990, using a mailed questionnaire to the 231 ASG member companies, addressed by name to the most senior information systems executive. The survey was carried out with the support of the Executive and administration of ASG.

A response rate of 61% was achieved which is very high for a mailed questionnaire (128 responses from the 210 full members of ASG with inhouse installations). Researchers involved in comparable USA studies have noted that a response rate of 36% is high and considered very satisfactory (Zmud, Boynton and Jacobs, 1987). A key factor in the high Australian response rate is the assistance and support of ASG, which is gratefully acknowledged. Such support means that the findings of the survey are more likely to be representative of ASG members and thus more useful as indicators of concerns and trends.

SCOPE OF THE SURVEY
The survey focused on eight major areas:
1. Organisational Profiles
2. The Information Systems (IS) Group in the Organisation
3. Information Systems Managers (ISM)
4. Information Systems Decision Priorities
5. Evaluation of Information Systems
6. Software Applications Packages
7. Organisational Information Systems Trends
8. Skills Levels and Requirements

While some questions in the survey were common to previous surveys, the last three of the above areas were covered much more fully in the 1990 survey. The purchase and implementation of software application packages has been an area of great growth in the last few years, yet there has been very little evaluation of the decision-making processes and evaluation of the implementation factors and outcomes of such purchases.

Questions related to Information Systems and Technology (IS/T) trends sought to identify the likely importance of certain developments to organisations. These were matched by current and required skills information to assist organisations, suppliers and industry bodies to determine future education and training needs.

The conduct of the survey and the reporting of the findings is aimed at achieving three major objectives: assisting organisations to compare the level of their own performance against industry norms; identifying ways in which managers can successfully lead and manage their organisation's investment in information systems and technology; and indicating major 'skills gaps' so that these can be addressed with appropriate organisational, industry and tertiary institution programs.

ORGANISATIONAL PROFILES
Respondents to the 1990 survey covered New Zealand, most states of Australia and included one respondent from Asia. Figure 1 below shows the distribution by country and Australian state. The state of NSW has the largest percentage of respondents, with 36%, followed by Victoria with 19%. New Zealand accounted for 12% of the respondents. There were no respondents from the Northern Territory.

<table>
<thead>
<tr>
<th>Location</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>New South Wales</td>
<td>36%</td>
</tr>
<tr>
<td>ACT</td>
<td>6%</td>
</tr>
<tr>
<td>Victoria</td>
<td>19%</td>
</tr>
<tr>
<td>Queensland</td>
<td>6%</td>
</tr>
<tr>
<td>South Australia</td>
<td>13%</td>
</tr>
<tr>
<td>Western Australia</td>
<td>7%</td>
</tr>
<tr>
<td>Tasmania</td>
<td>1%</td>
</tr>
<tr>
<td>New Zealand</td>
<td>12%</td>
</tr>
<tr>
<td>Asia</td>
<td>1%</td>
</tr>
</tbody>
</table>

Of the seven industry categories on the survey form, the highest percentage from the one industry area was 20% from 'Banking or lending'. The position of the smallest industry group was shared between 'Mining and Petroleum' and 'Retail or distribution' each with 8% (see Figure 2). Shifts of more than three percentage points from the 1987 survey occurred in four of the seven categories. Industry areas with increased percentages were 'Banking or Lending' (from 10.7% to 20%) and 'Services including Transport' (from 13.7% to 17%). Those with decreased representation amongst the respondents were in the 'Insurance or financial services' industry (from 19.8% to 15%) and the 'Manufacturing and process industry' (from 22.9% to 18%).

Because of the relatively small numbers in some of these groupings and to assist with later analysis, these seven categories were condensed into three major industry groupings: the Manufacturing group (34% of the total), which incorporates 'Mining or petroleum', 'Retail or distribution' and 'Manufacturing and process industry'; the Services group (52% of the total) encompassing 'Insurance or financial services', 'Banking or lending' and 'Services including Transport'; and the Public Sector group (14%), as it was listed on the survey from as category 'g'. Though the Public Sector group was disproportionately small compared with the Manufacturing and Services groups, we regarded the Public Sector organisations as sufficiently different to keep this category separate. Subsequent analyses in this paper justify this decision. Indeed, when compared with the 1988 figures, the 1990 group of ASG members are more heavily oriented towards the more information intensive service industries (see Figure 3).

![Figure 2. Industry Groups.](image-url)

![Figure 3. Major Industry Groups.](image-url)
When the geographic locations are matched with these major industry groupings, the distribution of respondent's industry groupings in Australia and New Zealand can be observed (see Figure 4). As would be expected, the ACT has a high public sector representation. The smaller Australian states do also (other than Tasmania), however, there were no NZ public sector respondents. NSW has a higher percentage of firms in the Service sector than does its neighbour Victoria, which is more heavily oriented towards the Manufacturing group. This reflects the balance of industries and firm headquarters in those states.

Figure 4. Geographical Concentrations of Major Industry Groups.

<table>
<thead>
<tr>
<th>State/Country</th>
<th>Service</th>
<th>Manuf</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSW</td>
<td>58%</td>
<td>35%</td>
<td>7%</td>
</tr>
<tr>
<td>Victoria</td>
<td>46%</td>
<td>46%</td>
<td>8%</td>
</tr>
<tr>
<td>Tasmania</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Queensland</td>
<td>50%</td>
<td>25%</td>
<td>25%</td>
</tr>
<tr>
<td>South Australia</td>
<td>40%</td>
<td>33%</td>
<td>27%</td>
</tr>
<tr>
<td>Western Australia</td>
<td>44%</td>
<td>33%</td>
<td>22%</td>
</tr>
<tr>
<td>ACT</td>
<td>29%</td>
<td>0%</td>
<td>71%</td>
</tr>
<tr>
<td>New Zealand</td>
<td>60%</td>
<td>40%</td>
<td>0%</td>
</tr>
</tbody>
</table>

ASG members come from a range of medium sized to large organisations. The mean size of organisations by number of employees was 6,218. However, the median was much lower at 1,220. This would indicate that the mean has been skewed by some very large organisations and that many ASG members are in the middle ranges in terms of size.

INFORMATION SYSTEMS GROUP IN THE ORGANISATION

In nearly half of the organisations (48%), the IS Department is a functional group in itself, with the 'Management services' area being the next most likely grouping (22%) (see Figure 5). There were some industry differences in the organisational location of the Information Systems Department, with particular differences being largely between Manufacturing and the other two sectors. Over one-quarter (26%) of the Manufacturing IS Departments reported to the 'Finance' Department whereas this figure was 6% for Services and nil for the Public Sector. However, the reverse was true for the 'Management services' area: 27% of the Services industry IS Departments and 28% of the Public sector reported to 'Management services' while this accounted for only 12% of the Manufacturing group.

Figure 6. Size of IS Group.

<table>
<thead>
<tr>
<th>Number of analysts and programmers in IS Department</th>
<th>1988</th>
<th>1990</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 8</td>
<td>14%</td>
<td>14%</td>
</tr>
<tr>
<td>Between 8 and 24</td>
<td>33%</td>
<td>22%</td>
</tr>
<tr>
<td>Between 25 and 74</td>
<td>35%</td>
<td>34%</td>
</tr>
<tr>
<td>More than 75</td>
<td>18%</td>
<td>30%</td>
</tr>
</tbody>
</table>

Nearly two-thirds (64%) of organisations had more than 25 systems staff (see Figure 6). This is considerably higher than the 53% in the 1988 study. This might be explained by the larger percentage of firms in the Services group, where the numbers of IS staff are larger (see Figure 7). In the Manufacturing and Public sector groups, about one-quarter of the organisations have fewer than eight systems analysts and programmers. The figures are 26% and 22% respectively, with the comparable figures for the Services group being only 5%. However, in the Services group, 39% are in the '75 or more' category, while for Manufacturing this is 19%, and the Public sector, 17%. The increase in numbers may very well have represented a lagged peak in the business cycle as several large organisations have had significant information systems staff retrenchment and redundancy programs since the survey was conducted.

There has been a dramatic shift in the extent to which end-users are processing their own data, as Figure 8 indicates. A much greater percentage of employees are processing data using microcomputer, minicomputers or...
mainframe facilities, particularly in the Services and Public Sectors. This has been predicted to continue (Hunt and Dampney, 1990). Australia and New Zealand appeared to be fast reaching the situation described by Keen where PCs, workstations and department systems account for more expenditure than do central IS expenditures (Keen, 1991, p 144). These changes would seem to emphasise the changing role of information systems in organisations to greater integration into the fabric of the business.

INFORMATION SYSTEMS MANAGERS
The managers who head these groups are in very senior positions (see Figure 9). Over 80% reported that there was no more than one level between themselves and the Chief Executive Officer (CEO). This is slightly lower than the 1988 figure of 87% and may be a factor of the increased organisational size of the participant group.

Figure 9. Reporting Level of IS Managers.

<table>
<thead>
<tr>
<th>Reporting Level</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directly to CEO</td>
<td>31%</td>
</tr>
<tr>
<td>One Level</td>
<td>51%</td>
</tr>
<tr>
<td>Two Levels</td>
<td>15%</td>
</tr>
<tr>
<td>Three or more</td>
<td>2%</td>
</tr>
</tbody>
</table>

The previous appointment area of the IS Manager was included in the 1990 study to identify if there was a range of background emerging for IS Managers (see Figure 10). The previous appointment of over 70% (73%) of IS Managers was in the Information Systems area, with the remaining 27% being spread amongst five other categories. Considering some of the skills needs identified later in this paper, it is interesting to note that only one IS Manager's previous appointment was in the 'Marketing' area and this was a marketing experience in the Services group. In Hunt and Dampney's recent Delphi study, more than 75% of participants believed that by the mid-1990s, it would be widely accepted that 'information systems were too important to be managed by EDP experts' and that practically all Information Systems Managers would be required to have a 'general business background' (Hunt and Dampney, 1990).

Analysis of the background of IS Managers against the three industry groupings revealed some notable differences. After accounting for those whose previous appointment was in the IS area, IS Managers in the Services group were likely to have experienced positions in the 'Finance or administration area' (11%) or 'External consulting'. In the Manufacturing group, experiences in 'Management Services' (14%) or 'Manufacturing or Production' (9%) were the next most likely categories. For the Public sector, of whom 83% came directly from the IS area, the remaining 17% comprised 11% from the 'Finance or administration' area and 6% from the 'Manufacturing or Production' area.

INFORMATION SYSTEMS DECISION PRIORITIES
The need for Business Managers to have greater ownership of IS developments is seen as a positive factor in the effectiveness of organisational information systems (Johnston and Carrico, 1988). In this study, we sought to identify who is finally accountable for their return on investment on an information systems application. Figure 11 shows that, predominantly, this is a joint I/S and Business manager responsibility, but there are some very clear industry differences.

Figure 11. Accountability of IS/T Return on Investment.

<table>
<thead>
<tr>
<th>Accountability of IS/T Return on Investment</th>
<th>Services</th>
<th>Manuf</th>
<th>Public</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint IS and Business Managers</td>
<td>67%</td>
<td>60%</td>
<td>50%</td>
<td>63%</td>
</tr>
<tr>
<td>Business Managers</td>
<td>27%</td>
<td>23%</td>
<td>44%</td>
<td>28%</td>
</tr>
<tr>
<td>IS Management</td>
<td>6%</td>
<td>16%</td>
<td>6%</td>
<td>9%</td>
</tr>
</tbody>
</table>

Ensuring a return on IS investments is much more a Business Manager responsibility in the Public sector than in private sector firms. This is confirmed in subsequent responses concerning responsibility for initiation software selection (see Figure 14) though curiously not for the subsequent software championing. These Public sector differences are explored further below.

IS Managers indicated that the systems development priorities of their CEOs was clearly towards 'Increased market share' followed by efforts to reduce costs. The overall rankings appear in Figure 12 below. Over 60% of the Services group and 56% of Manufacturers ranked 'Increased market share' first. The different Public sector environment was evident here in that this response was ranked first by only 33% of the Public sector respondents. Over 53% of this sector ranked 'Reduced costs' as the most important criterion for CEO support, and a further 18% ranked 'Control and monitor costs' the highest.

Overall, the results of systems development priorities indicated by Figure 12 support the changing emphasis in the role of information systems in organisations since these
surveys began in the early 1990s. There has been an increasing importance placed on the role of information systems and technology in providing systems and services which extend business opportunities, rather than those which seek to contain costs as their major rationale. This is particularly marked in the private sector and the importance attached to this in the Manufacturing area is noticeably increased over the 1988 data.

EVALUATION OF INFORMATION SYSTEMS

Information Systems Managers were asked what ratings they believed their (internal) clients would give them. These are listed in Figure 13 below and show a somewhat more positive perception than was anticipated by the researchers.

‘Client satisfaction on completion of major projects’ was rated most highly, with ‘Availability of operational data’ for querying by non-IS staff being seen as the poorest performer of the four. With the large increase in non-IS staff processing their own data, there is obviously an as yet unmet demand for IS Departments to make data more widely accessible throughout their organisations.

<table>
<thead>
<tr>
<th>Service</th>
<th>(%)/Poor</th>
<th>Fair</th>
<th>Satis</th>
<th>Good</th>
<th>Exc</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Systems Delivery</td>
<td>4</td>
<td>29</td>
<td>28</td>
<td>36</td>
<td>3</td>
<td>3.1</td>
</tr>
<tr>
<td>b. Availability of Data</td>
<td>20</td>
<td>19</td>
<td>21</td>
<td>34</td>
<td>6</td>
<td>2.9</td>
</tr>
<tr>
<td>c. Client Satisfaction</td>
<td>0</td>
<td>9</td>
<td>26</td>
<td>54</td>
<td>11</td>
<td>3.7</td>
</tr>
<tr>
<td>d. Modifying Systems</td>
<td>6</td>
<td>18</td>
<td>32</td>
<td>36</td>
<td>7</td>
<td>3.2</td>
</tr>
</tbody>
</table>

For the first time the 1990 survey included questions about software purchase options vis-a-vis developing software internally. This question also sought information concerning the preferred development location of purchased ‘Line of Business’ software applications packages. As Figure 16 shows, assuming functionally equivalent system options with the same payback period, ASG members had a strong preference for purchasing packages, rather than developing them internally. Packages developed in Australia and New Zealand were overwhelmingly preferred to those developed elsewhere. This preference should be of great interest to the Australian software industry and to those seeking support for that industry.

SOFTWARE APPLICATIONS PACKAGE SELECTION

Over 90% (93%) of survey participants indicated that they had acquired and installed a major application package from external sources and answered the series of questions about the selection and evaluation of software packages.

Cooperative efforts in the initiation of the search for software packages is clearly the norm amongst ASG members with two-thirds indicating IS and Business managers did this jointly. The overall figures though mask considerable differences amongst the three industry groups, with the Public sector again diverging from the responses of the other two groups (see Figure 14). The initiation of the software search is much more a Business manager than joint responsibility in government depart-

| Joint IS and Business Managers | 73% | 65% | 37% | 66% |
| Business Managers             | 8%  | 5%  | 37% | 11% |
| IS Management                 | 19% | 30% | 25% | 24% |

ments and authorities, though this is not the case for justifying the acquisition of packages.

The acquisition of particular packages is also predominantly a joint IS and Business manager responsibility, with Business managers having a greater responsibility as champions than in the initiation of the software search (see Figure 15). Hunt and Dampney’s study predicted that by the mid-1990s, computer applications development would be a largely user driven responsibility (Hunt and Dampney, 1990).

The Public sector organisations were less likely to develop packages internally (19%) and thus more likely to seek Australian and New Zealand developed packages (75%). However, there remained 6% of the Public sector firms who indicated a preference for software-developed outside those two countries.

Over 50% (55%) of respondents indicated that it was either ‘Very important’ or ‘Extremely important’ for purchased applications packages to conform to the corporate data model of their organisations (see Figure 17). By contrast, only one third of respondents (33%) regarded con-

<table>
<thead>
<tr>
<th>Level of Importance (Not &gt; Extremely)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
</tr>
<tr>
<td>a. Open Systems Standard</td>
</tr>
<tr>
<td>b. Your Corp Data Model</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>17 21 29 22 11</td>
</tr>
<tr>
<td>7 15 24 38 14</td>
</tr>
<tr>
<td>2.9 3.4</td>
</tr>
</tbody>
</table>
formity with open systems standards (as opposed to proprietary systems standards) as of such great importance.

**EVALUATION OF PACKAGES**

Despite the substantial sums of money and human effort invested in the purchase, implementation and exploitation of applications packages, there has been minimal external and industry evaluation in this area. This study sought to rectify this with a series of questions aimed at gaining the feedback about vendors and the implementation process from ASG members.

The accuracy of vendor presentations and level of vendor installation support were judged to be 'Good' to 'Excellent' about one third of the time and 'Poor' to 'Fair' just under 30% of the cases (see Figure 18). IS Managers in the Services industry grouping were more critical of vendors in these areas than were those in Manufacturing and Public sector organisations.

Vendors' post-installation support and the accuracy of their implementation timetables were ranked as 'Good' or 'Excellent' by only about one quarter of the participants, with 43% and 47% respectively indicating that they rated their vendors only 'Poor' or 'Fair'. Manufacturing sector firms tended to be more generous in their assessments than those in the Services and Public sectors.

**Figure 18. Vendor Evaluation.**

<table>
<thead>
<tr>
<th></th>
<th>Poor</th>
<th>Fair</th>
<th>Satis</th>
<th>Good</th>
<th>Exc</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Accuracy of Presentation</td>
<td>5</td>
<td>24</td>
<td>38</td>
<td>29</td>
<td>4</td>
<td>3.0</td>
</tr>
<tr>
<td>b. Level of Installation Support</td>
<td>14</td>
<td>23</td>
<td>31</td>
<td>29</td>
<td>3</td>
<td>2.8</td>
</tr>
<tr>
<td>c. Post-Installation Support</td>
<td>21</td>
<td>22</td>
<td>34</td>
<td>21</td>
<td>2</td>
<td>2.6</td>
</tr>
<tr>
<td>d. Accuracy of Timetable</td>
<td>23</td>
<td>24</td>
<td>28</td>
<td>23</td>
<td>3</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Considering the somewhat unsatisfactory performance of vendors as indicated above, there could be grounds for some IS staff resentment or a different decision taken with hindsight. However, this was not the case, as Figure 19 shows. Despite the problems, IS managers believe that the purchasing of the application packages was the correct decision and was not resented by IS staff. In this instance, a greater percentage of Public sector firms were in the 'Not sure' through to 'Strongly agree' categories, indicating perhaps more disappointing experiences or else a greater desire to develop packages internally.

A majority of the participants (59%) reported that the implementation of the applications package did not bring about any changes in the organisation's information systems environment. Manufacturing sector firms experienced the least change in their IS environment with 72% indicating 'No change'.

**Figure 20. Changes to the IS Environment.**

<table>
<thead>
<tr>
<th>Type of Changes</th>
<th>Extent of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>To Vendor's Package</td>
<td>Major</td>
</tr>
<tr>
<td>28%</td>
<td>40%</td>
</tr>
<tr>
<td>To Business Processes</td>
<td>13%</td>
</tr>
</tbody>
</table>

Nearly 60% of the ASG members reported that their most recent applications package was from North America, with 32% being Australian or New Zealand developed software (Figure 22). Despite their professed preference for local software as indicated above, the Public sector had the highest percentage of software purchased outside Australia with 75% from North America, 8% from Europe and only 17% from Australia and New Zealand. The rationale for the difference between professed and actual sourcing of packages is unclear. Comparable Australian software might not be available or, alternately, the survey participants provided the answer that they believed they should, rather than one which reflects actual behaviour.

A very high percentage of all industry groups were using Systems 370 Architecture as the operating system.
for the Package, reflecting the group targeted by the survey. The Public sector group had the highest non-IBM environment, with over 20% being listed as in the ‘Other’ category, and 80% in System 370 Architecture.

The findings in this study would indicate that ASG members using purchased application packages are experiencing fewer changes and less disruption to their organisational and IS environment than might be expected. In over 70% of firms (72%) only minor or no changes were made to vendor packages to suit organisational needs and only 13% of participants made major changes to their business processes to accommodate the implementation of the package.

**ORGANISATIONAL INFORMATION SYSTEMS TRENDS**

Business, industry and government organisations in Australia and New Zealand are in the midst of considerable technological, economic and social changes. After discussions with both Business and IS managers, and a review of concerns in other countries, the researchers compiled a list of twelve areas which were likely to be important to ASG organisations in the next decade. These were listed in the survey form and participants were asked to indicate the likely importance of each to their organisation in the 1990s (see Figure 24). Of the top five areas seen as key challenges, four reflected a concern for business rather than technically oriented issues.

These results indicated a high need for business and human communication skills to work with business managers, to use business oriented techniques and processes, to raise the IT awareness of managers and to meet the data access needs of business professionals. These areas of importance mesh with Silk’s data, that the organisational impact of IT/IS remains a key concern as does the problem of justifying investment in IT/IS (Silk, 1990).

The area where increased technical skills were perceived to be important for the 1990s was ‘Communication (Network) Management’. As organisations establish electronic linkages with each other, coincident with the availability of a much expanded message bandwidth, the need for an efficient and cost effective network is heightened. When the areas are analysed by industry, some sharp differences are noticeable. Both the Services and Manufacturing groups indicated that the ‘IS and Business partnership’ would be critical, with combined percentages for ‘Very important’ and ‘Extremely important’ being 86% and 81% respectively. The information intensive nature of the Services group and the strategic importance of information systems and technology to firms in that area was emphasised by generally higher levels of importance indicated in a greater number of areas than was evident in the other two groups. Other areas of high importance to the Services group were ‘Determining the return of IS’ (73%), ‘Executive decision support systems’ (65%) and ‘Systems delivery’ (63%).

The Public sector group indicated lower levels of importance across most of the business-oriented areas, other than the need for ‘Information Technology awareness of Business managers’ (67%). As with the other two industry groups, the area of greatest importance to Public sector organisations was ‘Communications (Network) Management’ (89%). These differences are important to hardware and software suppliers, consultants, industry and professional groups and to education and training providers.

**SKILLS LEVELS AND REQUIREMENTS**

Current and required levels of skills for both senior IS Managers and IS professionals were sought as part of this study. Participants were given a list of ten skill areas and asked to indicate both the current and required skill level, on a three category scale (Low, Medium, High) in their organisations.

Figure 25 lists those areas, showing the percentage of respondents who marked the current skill level in their organisation as ‘High’, followed by the ‘High’ category for the required level, then the percentage difference and
required business skills of information systems managers oriented, indicating a large gap between the current and listed first.

Again, four of the top five categories are management oriented, indicating a large gap between the current and required business skills of information systems managers and professionals. 'Aligning business and IS strategies' is clearly the most critical skill for all organisations, with high required skills ranging from 94% in the Services group to 78% in the Public sector. This is quite consistent with the findings of a recent study of 150 US-based information systems managers, where two of the top three concerns for 1990-95 related to the need for business oriented staff and a better understanding of business missions (Hershey and Eatman, 1990).

In this Australasian study, the skills gap is greatest in the Public sector as this group currently has the lowest base of current skills. However, respondents in this sector indicated lower levels of required skills in most areas when compared to their colleagues in the Services and Manufacturing organisations.

ALIGNING STRATEGIES
We were not surprised to find that the alignment of business and information systems strategies was identified as a key 'skills gap' area. In the 1988 survey the need to 'align information systems and business goals' was the highest ranked concern of IS Managers. Similar findings resulted from an Australian Delphi study reported in 1989, where 'improving IS strategic planning' was listed first in the top ten issues for Information Systems methodologies (Watson, 1989).

These Australasian results are similar to the findings in both the UK, Ireland and the USA: Galliers compared information systems planning practices in the United Kingdom and Australia in 1985 and 1986 and reported that very little of current practice then incorporated competitive considerations (Galliers, 1988). 'Information systems planning' has been consistently identified as the major issue of concern in recent US-based studies (Hartog and Herbert, 1986; Brancheau and Wetherbe, 1987). Concerns of chief executives in Ireland related to the management of information technology included the linking of information technology and business strategies (Moynihan, 1988).

What steps can business and information managers take to better align business and information strategy? This question is further explored in a recently completed study by one of the researchers which examined the information-based comparative advantages gained by some of Australia's largest financial institutions (Broadbent and Samson, 1990; Broadbent and Weill, 1991). A number of positive indicators for business and information
strategy alignment emerged from that study which help
answer this question. These focussed on the nature of the
firm’s organisation-wide strategy formulation and imple-
mentation process, the roles and responsibilities of busi-
ness and information managers and the interaction
between and experiences of business and information
managers.

A major reason why IS managers experience difficulty
in keeping pace with business requirements is that it is
often hard to clearly identify a firm or government
department’s business strategy. Such a strategy might be
expressed in a form which requires much more elaboration
for appropriate systems developments. An overriding find-
ing to emerge from the alignment study was that in order
for there to be information and business strategy align-
ment, the direction and focus of a firm’s corporate and
business strategies, and hence developmental priorities,
need to be clear and to be understood (Broadbent and
Samson, 1990). It appears that without that prerequisite,
inappropriate choices are more likely to be made in the
development and implementation of information strategy
and thus mis-alignment more likely to evaluate.

MARKETING AND PARTNERSHIPS
Linked with ways of improving the alignment of business
and information strategy is the clearly identified need for
better marketing of information systems services. Our var-
ied experience of working with information systems man-
agers raised concerns about IS Managers’ understanding of
the differences between marketing and promotion. We
hope that the response to the need for a much higher level
of marketing skills was not interpreted simply as ‘the
promotion of IS services’. Marketing encompasses identi-
fying purposes and needs, and then developing services
and/or products to meet those needs. ‘Promotion’ is but
one small part of that process and comes in the later stages.

Marketing of information systems and services within
organisations requires sound diagnostic skills. Business
users often cannot clearly and accurately articulate what it
is they want or need in the information systems and tech-
ology area. They often do not know what is possible, or
even reasonable, and thus IS managers and personnel need
to develop a consultancy approach within their own organi-
sation (see for example Arnoude, Ouellette and Whalen,
1988).

The need for greater marketing skills is linked to the
high ranking (number 2) given to ‘IS and business partner-
ship’ as an area of high importance amongst ASG member
organisations in the 1990s. Throughout this survey, in a
number of questions, consistently high rankings were
given to responses which linked ‘IS and business managers’.
Our observation is that there is much more interaction
between business and information managers than even two
years ago. This is partly due to the ‘federated’ nature of the
information services groupings in many organisations, par-
ticularly large and multidivisional firms. In a federated
arrangement, IS is largely a distributed function, with each
business unit containing and largely controlling its own IS
capability. A central IS unit has responsibility for defined
aspects of policy and architecture across the organisation
and might deliver some common or shared services
(Feeny, Earl and Edwards, 1989).

CONCLUSIONS
This study has identified issues of critical importance to IS
managers and highlighted areas where skills need to be
further developed. The rise in the growth of end-user
computing raises questions concerning the appropriate
management of devolved or diffused computing arrange-
ments and access to organisational data.

There is an expectation by business managers and users
of ready access to organisational data. This access is
increasingly being sought, though not realised, by
devolved computing arrangements. As Keen has noted,
departmental computing assets are often under-managed
and the cost of their support underestimated (Keen, 1991,
p 148). Organisational costs such as the management time,
the learning curve for staff and the costs of transition to
new systems are rarely budgeted appropriately. The more
appropriate management of devolved computing, together
with adequate access to and exploitation of organisational
data would seem to provide fruitful areas for realising a
greater return on information technology investments.

Organisations in this study showed a strong preference
for purchasing software packages rather than undertaking
inhouse developments. The incorporation of packages in
the systems development process now appears well estab-
lished. The preference for Australasian software packages
would seem to provide a window of opportunity for local
software houses.

The strength of the business and IS partnership has
increased since the 1988 study, as has the strategic impor-
tance of information technology to ASG member orgnisa-
tions. However, the ability to align business and informa-
tion strategies and the marketing of information services
remain high on the list of skills gaps. The results of the
1990 survey further emphasised the importance of the
management and business functions and skills required of
IS managers.

In the 1990s there are many more ‘stakeholders’ con-
cerned with information systems and technology than in
earlier times. The changing responsibilities of managers
has implications for the education, training and experienc-
es required of both business and information managers.
Today’s business managers require a much greater under-
standing of technology than did their predecessors. At the
same time, information systems managers who wish to
make an impact in their organisations require business
skills and a focus that have not been part of the education
of technologists.
This gap now appears to be recognised by tertiary institutions. Recently developed business-oriented courses aimed at middle level IS professionals include the Master of Management (Technology) offered by the University of Melbourne's Graduate School of Management and RMIT's Master of Business (Information Technology).

The past decade has seen the development of many products and processes which have increased the potential strategic relevance of technology to organisations and heightened the importance of the information systems and services management in organisations. In this study, IS managers have continued to show their desire to meet the business and organisational needs of their firms, government departments and industries. However, renewed efforts are required to enhance the skills of information managers to meet the emerging demands of organisational and user expectations.

REFERENCES


ACKNOWLEDGEMENTS

The researchers appreciate the help of ASG members and the Executive Board for their cooperating in working on and completing the survey and would like to thank IBM Australia Ltd for the funding to undertake this project.

The principal author of this paper wishes to acknowledge the support of two organisations with which she was associated during the data collection process for the study: first, her previous employer, Royal Melbourne Institute of Technology; and second, the Centre for International Research in Communications and Information Technology (CIRClIT) where she worked on secondment as a Research Associate for three months in 1990.

BIOGRAPHICAL NOTES

Dr Marianne Broadbent is Program Director of the Key Centre for Strategic Management at the University of Melbourne's Graduate School of Management, 200 Leicester Street, Carlton, Victoria, Australia, 3053. She was previously Head of the Department of Information Services at the Royal Melbourne Institute of Technology. Her research interests focus on aligning business and information strategies, business process redesign, the management of information resources and technology, and the education and training requirements of both general and information managers. She is Program Coordinator of the Master of Management (Technology).

Philip Lloyd is an Information Systems consultant with a particular interest in the effectiveness of the management of information and information technology within organisa-
Alan Hansell is a Senior Systems Representative with IBM Australia. He has worked for IBM for 28 years in the information systems area. His research interests are in state-of-the-art systems development, and executive reporting for IS matters.

Associate Professor C.N.G. (Kit) Dampney is based in the Computing Discipline at Macquarie University. He has previously held joint appointments in Macquarie University’s Graduate School of Management and with the CSIRO’s Division of Information Technology as Senior Principal Research Scientist. His interests focus on bringing principle and practice together from the strategic context of information systems through to building them. He runs the Macquarie University post-graduate programme in Computing and Information Systems.
Object-oriented information systems: An introductory tutorial

B. Henderson-Sellers
School of Information Systems
University of New South Wales

The use of object-oriented techniques in computing and information systems development is likely to increase over the next few decades to become a new standard alongside structured techniques. Consequently information scientists and computer technologists need to become conversant with at least the basic ideas of the object-oriented philosophy. Those ideas are presented here in a form independent of the language of implementation. Examination of the underlying three concerns of encapsulation/information hiding, abstraction/classification and polymorphism/inheritance leads to consideration of the terminology of objects, classes and class-class interactions.

CR Categories and Subject Descriptors: 0.2.10 (Software Engineering): Design — methodologies, representation.

General Terms: Design, notation, information systems.

Additional Keywords and Phrases: Object-oriented software engineering concepts, notation.

Copyright © 1992, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that the ACJ’s copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society Inc.


1 INTRODUCTION

There have been many introductory articles on the object-oriented (OO) paradigm/philosophy; yet these have concentrated largely on OO as a method of programming (eg Rentsch, 1982; Howard, 1988; Stroustrup, 1988; Pokkunuri, 1989). Indeed the recent interest in OO has been 'driven' largely by the availability, over the last four or five years, of good object-oriented programming languages and associated programming environments. It is only over the last 1-2 years that a concomitant development has arisen in the application of OO ideas to information systems and hence to earlier phases of the software development lifecycle (loosely termed here analysis and design) as well as to the dynamics of the full software lifecycle (eg Henderson-Sellers and Edwards, 1990).

In this paper, we give an introduction to object-oriented concepts, as applied throughout the software development lifecycle. The ideas of objects, object representation and object-object interactions form a “running theme” to this paper. Since the underlying conceptual model is consistent across all phases of the lifecycle, it is consequently easy to discuss this common model at different stages and yet this very conformity sometimes presents difficulties to the novice since the traditional distinction between the various lifecycle phases becomes highly blurred.

Since the same model of an "object" is used throughout the various phases of the lifecycle, the overall approach is often referred to as a “seamless transition” from analysis, through design to programming. This is in contrast to a procedural decomposition where you often use data flow diagrams and perhaps hierarchy charts in analysis and/or design and then have to translate these two very different models into a third model for detailed design and coding; taking into account the strictures imposed by the procedural language of your choice as you start to code.

The use of a single model throughout the lifecycle means that you can use essentially the same words because you are handling the same basic idea throughout. Conversely, if you are using the same words, ie using the word ‘object’ for design objects, analysis objects and run-time objects, there may be a problem of identifying readily the relevant stage of the lifecycle. In general, this is not so obtuse as it might at first appear because any discussion often pertains primarily to one particular stage and the meaning of the word ‘object’ becomes obvious.

Nevertheless, some authors do attempt to differentiate lifecycle stages by modifying their ‘object’ terminology. Some common terms are shown in Figure 1. At the analysis and early design level, you hear people talk about objects or entities (see eg Bailin, 1989; Korson and McGregor, 1990). In the design phase, the phrase ‘object classes’ (which are classes, sets or collections) is often used; whereas the term ‘classes’ refers to coding, and ‘run-time objects’ refers to individual instantiations (or instances). Others prefer to underline the similarities rather than
What is an Object?

Object = Data + Functionality + Encapsulation

<table>
<thead>
<tr>
<th>LIFE CYCLE STAGE</th>
<th>NOMENCLATURE</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis and Design</td>
<td>Objects or Entities or Object Classes (O/Cs)</td>
<td>Bank Accounts</td>
</tr>
<tr>
<td>Design</td>
<td>Classes</td>
<td>Code describing all personal bank accounts</td>
</tr>
<tr>
<td>Run-time</td>
<td>Object</td>
<td>ADT describes 'Account' type: Interface and features but not implementation</td>
</tr>
<tr>
<td></td>
<td>object</td>
<td>One specific bank account, belonging to one specific person</td>
</tr>
</tbody>
</table>

Figure 1. Object/class nomenclature throughout the lifecycle.

the differences: for example, ‘object classes’ (Meyer, 1988) or its abbreviation ‘O/C’ (Henderson-Sellers and Edwards, 1990); or ‘Classes-&-Objects’ (Coad and Yourdon, 1991).

In the remainder of this paper, we use the terminology of O/C to represent the underlying object model which is realised as a ‘set’ concept in analysis, design and coding; but as an ‘individual instance’ concept when the program is executed (run). If we wish specifically to refer to code written (the class) or to the run-time instantiation (object), then those particular, lifecycle-specific terms will be used.

The use of a single name (O/C) emphasises the significant blurring between lifecycle stages (eg Henderson-Sellers and Edwards, 1990) making clear-cut distinctions between phases both difficult and unnecessary.

The right hand column of Figure 1 shows, as a brief example, a bank account which describes a set of all bank accounts at analysis and design. As you move from analysis through design to coding then this notion of collection persists. What you actually write in an object-oriented program are classes; in this example, code for bank accounts. The coded class is a template from which we can instantiate individual examples of that class

2 THE OBJECT-ORIENTED APPROACH TO SYSTEMS DEVELOPMENT

Object-orientation² is a way of thinking, not tied to any particular language, merely a mindset which is itself supported better by more recent languages. It is this new way of thinking that I shall recommend to you rather than language syntax. One of the better ways to develop this new mode of thinking about software is firstly to avoid becoming embroiled in ‘language wars’ or ‘notational wars’ (for example, the so-called ‘bubble wars’) and secondly to try to assimilate the underlying concepts of the paradigm. Consequently the emphasis in this paper is on understanding the object-oriented philosophy predominantly through analysis and design rather than concentrating on language syntax (cf. Wybolt, 1991); although some reference to languages is included for completeness. Once these ideas become ‘natural’, the learning of a language syntax becomes much easier (eg Waldo, 1990) — there are many excellent language specific texts on the market.

The object-oriented paradigm, at its simplest, takes the standard components of any software system: data and procedures, but de-emphasises the procedures, stressing instead the encapsulation, in an autonomous module, of data and procedural features together, exemplified by the clear and concise specification of the module interface. In a systems decomposition based on an object-oriented approach, the system is viewed as a collection of O/Cs. High level analysis and design is accomplished not only in terms of these O/Cs but also in terms of the ways in which O/Cs interact with each other via ‘messages’ which pass information, invoking the objects to implement a procedure (its ‘behaviour’) or to reply with details about its state. As with structured systems development, the lifecycle phase of analysis relates to the problem space and interaction with users; whilst design refers to deriving a (software) solution. However, in contrast to structured techniques, there is no translation of analysis techniques and notations, such as DFDs, to other, very different design terminology. Rather, the smooth transition relates to the addition of more detail as the developer transfers from the problem domain to the solution domain, whilst retaining the same aspects of the object-oriented approach to software engineering and information systems.

Detailed design, including procedure implementation and specification of data structures, is deferred until much later in the development process and details are generally private to the object, thus adhering strictly to the concepts of information hiding as promulgated by Parnas (1972). Consequently algorithmic procedures and data structures are no longer 'frozen' at a high level of systems design but rather described at the detailed level of class algorithm specification. A system based upon object-oriented representation can remain more flexible since changes at the implementation level are more easily accomplished. Implementation details tend to be hidden and therefore changes have highly limited impact on other parts of the program. It is important that data structures should not be specified too early in the design process. Data entities may, however, provide the basis of object identification around which an interface is then developed. Thus the techniques of data modelling can provide a significant input to O/C identification and development (see also Rumbaugh et al., 1991). Thus object-oriented development focusses on data abstraction rather than freezing specific data structures into the object specification.

In contrast to the common structured systems analysis, based largely on top-down functional decomposition, object-oriented design and analysis has many attributes of both top-down and bottom-up design. Since one of the aims of an OO implementation is the development of generic classes for storage in (reusable) libraries (the software engineering 'holy grail' of true reusability), an approach which considers both top-down analysis and bottom-up design simultaneously is likely to lead to the most robust software systems.

Since a significant portion of object-oriented systems development is bottom-up, the differentiation between program design and coding is much less distinct than in a procedurally based systems lifecycle (eg Meyer, 1989a). However at this later stage, it would seem reasonable that within individual code modules, called classes in an object-oriented system, the tools developed for high level functional decomposition and top-down system design, such as DFDs, can still be found to be useful. However, these are used to delineate much smaller pieces of code viz. internal object structure, rather than program-level, inter-module interactions. Other graphical tools which are useful at different stages within the OO systems lifecycle are object-relationship graphs, client-server diagrams, inheritance charts or collaboration graphs (Wirfs-Brock and Johnson, 1990; Wirfs-Brock et al., 1991).

3. THE OBJECT-ORIENTED TRIANGLE
Object-orientation is generally agreed (Figure 2) as being the synergistic embodiment of essentially three concepts: some notion of encapsulation and/or information hiding; abstraction and classification; and polymorphism as implemented through inheritance.

3.1 Encapsulation and information hiding
Encapsulation and information hiding relate to the modularisation of programs and the exchange of data across the interfaces between the modules. They are closely related concepts and often treated as roughly synonymous. For example, Thomas (1989) suggests that 'encapsulation is the technical name for information hiding'. On the other hand, some people try and differentiate them conceptually as one being an idea and the other an implementation of the idea. Perhaps more strictly, code can be encapsulated (viz gathered together in a code module) yet still possess a high degree of visibility through the use of global variables (eg Wirfs-Brock et al., 1990, p 18). In other words, encapsulation does not guarantee information hiding; although the reverse is essentially true, at least in OO, since if the code has a high degree of information hiding this can only have been accomplished through the use of weakly coupled modular structures with a high degree of internal logical cohesion which encapsulate the nature of the problem space O/C being modelled.

Consequently, in a good object-oriented system both information hiding and a high degree of encapsulation will be found. Modules will have small interfaces so that the data and procedure calls across that interface between modules can be very tightly and cleanly specified. Furthermore, if changes are made to the internal structure of modules, only those variables declared in the interface can exert any effect outside of their local scope within the module. Common concerns in procedural programming regarding the global scope of many variables can to a great extent now be eliminated in an object-oriented system.

3.2 Abstraction and classification
The second basic concept (Figure 2) relates to abstraction and classification. Abstraction relates to the mental faculty of understanding at varying degrees of detail; isolating those aspects of importance and suppressing those of little importance to the current problem (cf. Rumbaugh et al.,
Classification builds on the notion of abstraction — the ideas of grouping software ideas into classes of things with similar structure and behaviour. This is directly analogous to the use of most nouns in a written and spoken language. For example, the noun dog may be used by a speaker to describe a specific dog. But we also can understand the use of this word dog in a more general context. For example, in the sentence, 'A dog is a person’s best friend', the word dog relates to the class, set or collection of all objects which can be described by the word dog. Here the word dog is being used for classification purposes and thus in English (and other languages) the semantics of such a noun are overloaded, dependent upon context. In a software context, then, a piece of code can be written to represent a generic bank account, rather than needing to represent a single person’s bank account. Relationships between things can be examined and identified in terms of relationships between classes of objects rather than individual objects, using techniques formalised in data modelling; so that analysis, design and now coding can all be undertaken using such class-based languages and concepts. This extends the potential usefulness of an information system by ensuring its wider applicability to problem areas far removed from the original context.

The process of classification is that of identifying like objects into classes. For example, consider rosella, bell magpie, kangaroo and koala. Rosellas and bell magpies can be classified as ‘bird’ and koalas and kangaroos as ‘marsupial’ (Figure 3). In this case we can say that ‘a rosella is a member of the bird class’. Whilst such an abstraction is useful at one level, for other purposes more detail will be required. A well classified system thus permits further refinement by the use of generalisation and specialisation. For example, we might classify rosella (at a more detailed level than simply bird) as ‘parrots’. Parrots are a more specialised (and hence more restricted) class than ‘bird’, but the class ‘parrot’ inherits directly from (is a subclass of) the class ‘bird’ (Figure 4). A rosella is then an example of the parrot class. Specialisation is the process of creating new, more specialised subclasses from the existing class.

Generalisation can thus be regarded as a second type of abstraction process being the identification of commonalities between classes which leads to the creation of a ‘parent’ class or superclass. The class bird and the class marsupial in Figure 4 have a common superclass of animal (Figure 5). This type of abstraction allows characteristics already identified, say for parrot, which relate to its bird-like nature of, say, flight, to be immediately available to non-parrots by locating such shared behaviour in the parent class of ‘bird’. Consequently a description of flight, made in the original analysis, need not be ‘reinvented’ when a new subclass is added to the system. Generalisation is essentially the ‘opposite’ of the process of specialisation which creates inheritance hierarchies downwards; whilst generalisation can be thought of as growing inheritance hierarchies upwards. As can be seen from Figures 3-5, the processes of generalisation and specialisation link with inheritance (see next section).

3In this very simple example we have taken the extreme liberty of ignoring the existence of all flightless birds.
relevant characteristics); the idea of encapsulation is around (Modula 2 and Ada are highly encapsulated languages). Inheritance is simply a software analogue of a taxonomic inheritance in a biological sense. An inheritance structure is one of the ways of offering reusability, extendibility, lower maintenance cost and of achieving the software engineering goals we have been aiming at for 20-30 years.

With inheritance we have an O/C called bank account, that is, a design object which is a collection of all bank accounts. We can also identify sub-sets of bank accounts — for example, chequing accounts, saving accounts. Each of these subsets inherits information from the ‘parent’ banking account. In this taxonomic sense (Figure 6) you can think of bank account as a sort of parent, a chequing account as being a sort of child where a savings account or a chequing account is just like a bank account except it is slightly different (see further details in Section 4.2).

Inheritance thus provides a method of relating classes in a way which is semantically sound. Such relationships, which parallel taxonomies most closely, are evident throughout the OO life cycle. Additionally, in programmer application, inheritance comes to be viewed as a method of sharing code. It is a method of saying, ‘Here I have a piece of code representing bank accounts, and here I have a piece of code representing savings accounts’. This sub-group is a subset of all other bank accounts in the world. They’re much like all other bank accounts except they have an extra characteristic: they provide an interest rate to customers. In constructing savings accounts, I don’t need to rewrite all the code to describe what all bank accounts can do: provide withdraw, transfer and deposit facilities and inform the customer of their current balance, for instance, in order to describe savings accounts since the majority of the characterisation of ‘savings accounts’ can be inherited from ‘bank accounts’ in general. They simply need an additional characteristic to reflect how they differ from bank accounts in general viz an interest capability.

**4 OBJECT-ORIENTED JARGON AND CONCEPTS**

**4.1 What is an object?**

In Figure 1 was outlined a meaning (if not yet a definition)
of the key word ‘object’ and all its variants (especially O/C — used here). An O/C is essentially an encapsulation of data and functionality. When you run a program using O/Cs, you are describing individual examples: particular objects, particular people, particular bank accounts. However, when you do analysis and design, you are generally not interested in single things but in a more abstract notion of the set, or class, of all similar things, as noted above.

Finding the O/Cs is loosely based upon identifying the nouns (Abbott, 1983; Booch, 1983) in the requirements specification viz. the user’s description of the system. This usually contributes a ‘first pass’ for the top-level O/Cs in the system. These tend to be O/Cs identifiable by substantive nouns. However, abstract nouns are just as likely to be candidates for becoming O/Cs (eg Harmon, 1990). Indeed, as systems are analysed and designed, it is often those more abstract concepts that emerge during system refinement. During system development, these ‘first pass’ O/Cs (substantive and abstract) will be significantly augmented by O/Cs created as artifacts of creating the conceptual model.

If you are designing a bank account system, for example, you are not particularly interested in designing just your collection of particular bank account(s); you are more interested in designing something that is widely applicable for everybody’s bank account. Consequently, when you do the analysis and design you are interested in sets of things, in collections of bank accounts. During these phases you tend to draw your O/Cs, with whatever icons you are using (see Edwards and Henderson-Sellers, 1991), to represent a set of all O/Cs describing, for instance, bank accounts, bank teller, bank customer, automated teller machine (ATM) and so on. At run-time and execution we are interested in particular customers and particular bank sub-branches and particular teller machines and so on. So at run-time we are interested in O/Cs which are indeed objects in the sense of being individual instantiations: in other words instances.

On the other hand, the class is what you actually write as code. The class is thus the coded version of the design objects which therefore includes code to describe the data associated with the O/C as well as its functionality (or behaviour). In most OOPLs, the idea of a class and a user

classes and abstract data types are actually central to the object-oriented approach; the abstract data type essentially being a formal specification of the object class. Hence when you are designing, you are essentially designing the abstract data type.

For example, I may describe a bank account O/C as having characteristics of account number, account name, account type, current balance and so on — and functionality like withdraw, deposit, transfer, etc. At the abstract data type level, I am not interested in how these functionalities are accomplished. The withdrawal or deposit algorithms are hidden to me in my role as a customer using an ATM. Neither do I care about the details of such algorithmic implementation; nor whether my bank decides to change this implementation; so long as it doesn’t make any material changes that might affect my use of the ATM — or the current balance in my account! In the real world (including the software engineer’s world) we are really interested in the external appearance or the interface ie what we can see from outside. We are not interested in how things are accomplished, only that the functions of the interface are accomplished. In contrast, in functional decomposition you always start with ‘How does this thing do it?’, ‘What is the procedure?’, ‘What is the function?’ and then later you think of the data with which you can do it. In the object-oriented approach to software development you look at the thing and what it offers in terms of services (what it can supply to another O/C) and/or in terms of state (data) and behaviour (functionality): We can simply consider that the O/C is recognised by its external view, or specification, and all implementation is hidden away in a private part of the O/C (eg Woodfield, 1990). A major software engineering advantage here is that you can change the implementation without affecting the interface. This has significant repercussions for software reliability, for reusability, for maintenance and so on.

The definition of a run-time object is therefore an instantiation of a class, created as and when required. So you could have three bank accounts, myaccount, heraccount, hisaccount, which are all individual instantiations (objects) created from the single class template BANK-ACCOUNT. A bank account O/C at analysis and design refers to a set of things. The code is the class of ALL accounts, with features such as account number, account name, balance and so on. At run-time we have run-time objects which have very specific attributes (fields). Account number now has a very specific INTEGER value and the account name a specific character string representation.

A class is an implementation of an abstract data type. Therefore it is a widely applicable template with which to

⁵Of course if we changed the name to the class-oriented paradigm then we would lose all those lovely double-O acronyms — but then there are people who think that it would be ideal! (eg Hecht, 1990).
describe shared characteristics. Object-oriented systems comprise coded classes. Classes therefore exist at compile time. A class could represent all animals, all bank accounts, etc; whereas what exists at run-time are individuals. The class is the implementation of the ADT. The ADT describes the external view. Conversely the ADT is the class without the implementation. Indeed, since the ADT is the specification and the class the implementation of the ADT, it is in fact possible to construct more than one version of a class which implements an ADT (in the same way that, at a lower level, one of several algorithms could be implemented to provide a 'sort' functionality).

The set of messages to which the class responds is called the protocol. It essentially represents those methods defined in the class interface, together with some notion of contract or class responsibility.

Looking inside an object more closely, we typically talk about features, which are the characteristics of the interior. Many of the features correspond to services offered to other O/Cs; others are strictly private features. Some typical terminology is shown in Figure 7; although this terminology is language-dependent to a degree. Features may be attributes or methods (also known as routines or member functions). Adopting the word 'method' generically, then we can divide methods into functions, which query the state of the object, and procedures, which change the state of the object. At this stage we are now at the level of procedural code and can use much of the expertise we've developed over the years in functional decomposition and procedural coding.

However, the style of programming, even within a class, should be in sympathy with the overall aim of object-orientation. This does lead to the development of a new programming style, even in designing and coding internal procedural elements. The use of inheritance, at the code level, requires acquisition of new skills, as does the use of complex types and of instance variables, and of message sending. Unfortunately, such stylistic guidelines are not possible to demonstrate within the small examples that must be used here. Once the introductory learning phase is successfully complete, however, you should consult more advanced texts on object-oriented programming style and aim to learn these new skills, so useful for programming-in-the-large.

Attributes are essentially the variables of the class which represent the characteristics of the class (viz its state) rather than its behaviour. These may be simple data assignments or may be implemented by computation and/or reference to other objects. Often the data values are of a basic type such as INTEGER, CHARACTER, STRING etc., which, in most OOPLS, are themselves regarded as object types.

---

\[6\] Strictly, methods is the Smalltalk word, routine Eiffel and member function C++.
Classes and Subclasses

Multiple Inheritance

(a)

<table>
<thead>
<tr>
<th>CHEQUING ACCOUNTS</th>
<th>Passbook Accounts</th>
<th>Savings Investment Accounts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streamline Accounts</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b)

Figure 9 (a). Venn diagram representation and (b) inheritance tree representation of multiple inheritance for the bank account example of Figure 6 extended by the addition of a new account.

As noted in Section 3.3, an alternative representation for inheritance is the inheritance tree (Figure 6). Frequently terminology of classes, subclasses and superclasses is used to describe the relative nature of the classes. The base class (superclass) here is BANK ACCOUNT, with two derived classes CHEQUING ACCOUNT and SAVINGS ACCOUNT. In this example, we have only included subclasses from one class, SAVINGS ACCOUNT. The whole nomenclature of subclass, class and superclass is, of course, relative and in any inheritance structure it is likely that there will be many more ‘levels’ than these three. To accommodate this, Meyer (1988) introduces names which permit differentiation between one ‘generation’ removed (up as parent and down as heir) and more than one generation removed (up as ancestor and down as descendant). He also notes that this avoids the apparent contradiction in the use of the word subclass since this implies a subset. Although this is true in an inheritance sense, from a module perspective derived classes are seen to extend the base class, since the services available to the subclass can be greater in number than those available in the class itself (Meyer, 1988, p 233). Note that if streamline accounts, which are essentially chequing accounts combined with interest-bearing savings accounts, are added to this example (Figures 6 and 8), then the Venn diagram (Figure 9(a)) becomes even more difficult to draw and/or comprehend, whilst in the inheritance tree (Figure 9(b)) a subclass (here Streamline Account) is added which inherits from more than one ‘parent’. This is ‘multiple inheritance’ and is supported in many, but not all, OOPs. Rather than a tree structure, as used to describe a single inheritance hierarchy, a multiple inheritance hierarchy is better described as a network or lattice.

One potential problem with such multiple inheritance (MI) is the potential for a clash of names of inherited features. In principle, the subclass inherits every feature, although in a hybrid language C++, for example, there are other choices. Resolution of name clashes of a pair of features (one from each parent class) is language-dependent.

Within an inheritance hierarchy, it is likely that some (or all) of the topmost classes may contain features whose definitions are deferred to the subclasses. In other words, there are no implementation details for these features within the base (parent) class. This means that this uppermost class cannot be instantiated and this type of class is subsequently known either as a deferred class (eg Meyer, 1988, p 234) or as an abstract class (eg Goldberg and Robson, 1983; Booch and Vilot, 1990; Lippman, 1989, p 304; Wirfs-Brock et al., 1990). The design and use of abstract classes is discussed by eg Johnson and Foote (1988); Wirfs-Brock and Johnson (1990).

One particular type of abstract class (viz. one will never be instantiated per se) is the ‘mixin’ (eg Bobrow, 1989; Booch and Vilot, 1990). This is a class which is probably not a full ADT but rather an encapsulation of several closely associated features bundled together as a class available to be ‘mixed in’, via multiple inheritance, to add capabilities to a wide variety of classes, possibly not related. The behaviour of the mixin is seen as orthogonal to the behaviour of the classes with which it is to be combined (Booch and Vilot, 1990), insofar as they add in ‘sideways’ extra characteristics which does not therefore require all classes in the inheritance hierarchy to support the mixin behaviour. Bracha and Cook (1990) view a mixin as an ‘abstract subclass’ — for example a subclass which might add a border to a wide variety of window classes.

Repeated inheritance occurs when two of the classes from which you inherit, themselves inherit from a common ancestor. In this case, it is likely that two features, one inherited from each parent, will have originated from a unique ancestral feature, possibly with redefinition in the lineal descent.

For example, in terms of bank accounts (Figure 6), you define common information relevant to all bank accounts in the topmost class, eg withdraw, deposit algorithms, balance. (Here I am using top in the sense of top of the page in the appropriate cited figure which is essentially the most general, base class — a convention also used by Wirfs-
As noted above (Figure 6) some bank accounts earn interest (and can be classified as Savings Accounts) and some don't (Chequing Accounts). Using inheritance we can say chequing account inherits everything that is known about bank account, eg it has a balance, you can withdraw money from it (and similarly for savings account); but, in addition, there are some special rules for chequing accounts and savings account: you cannot write a cheque on a savings account. These business rules are, then, very specific to certain sub-collections of bank accounts. Thus the higher level ('parents' and 'ancestors') classes tend to be more abstract than the 'children' and 'descendant' classes, which are much more specific (Harmon, 1990). Savings accounts can also have 'different flavours'. For instance, a saving investment account will be just like a passbook account except it has a different minimum period of investment, a different interest structure etc. and both are subsets of all savings accounts. In a savings investment account you probably have to keep your money in the bank for a certain period of time before you earn interest (at a higher rate), but otherwise all the other rules hold. The code about the bank account will have information about things that are common to all these subsets of different sorts of bank accounts (chequing and the various types of savings accounts), the code for savings account will modify code with rules about interest, and the code for savings investment accounts is likely to include also rules on minimum deposit periods.

Furthermore, when you design a new account, let’s call it a ‘super savings account’ which is just like a savings account except . . ., you add it to the design of Figure 6 by adding a third ‘child’ to the savings account ‘object’ (Figure 10). The coding of that (often called ‘programming by difference’ eg Johnson and Foote, 1988) will be about four lines of code which says something along the lines of ‘super savings account is just like a savings account except that you have (say) two extra features’. In contrast, the super-parent bank account object may be 20-30 lines long delineating methods for transfer, withdraw etc. which can be inherited by all its descendants (all the other objects in Figure 10). That is a great contribution to reusability. Finally, if the new class inherits from multiple parents, as in the STREAMLINE ACCOUNT in Figure 9(b), then repeated inheritance is also possible. In this example, class STREAMLINE ACCOUNT is seen to inherit from BANK ACCOUNT by two different routes. (For further details and design examples, see Henderson-Sellers, 1991).

4.3 Object-object interactions (client server model and software contracting)

There are three types of inter-object relationship generally recognised in the analysis phase (Loomis et al., 1987): aggregation, association and inheritance. Aggregation represents the has-a or consists-of relationship (eg a room consists of four walls, a floor and a ceiling); association is the direct use of services of one object by another (eg a customer object uses the services of a bank object) and inheritance represents a taxonomic hierarchy, also known as an is-a relationship. These three relationships can be thought of as being, in some sense, mutually orthogonal so that quasi-three-dimensional diagramming techniques are required. Although most OO languages do not differentiate between aggregation and association, such a differentiation has been found to be useful in analysis and broad (early) design. Currently the OO programming model supports only a client-server relationship.

At coding there is no differentiation in any of the current OOPs between aggregation and association. Currently the client-server relationship simulates both association and aggregation (although in some languages nested O/Cs or expanded types can be used to model aggregation). Consequently, at the language level, objects essentially interact in only two ways: inheritance and the client-server mechanism (Wirfs-Brock et al., 1990) or client-supplier (Meyer, 1988). This latter interaction forms the basis of viewing the overall design approach as being responsibility-driven — a term coming from Rebecca Wirfs-Brock's work (Wirfs-Brock and Wilkerson, 1989). Employing this type of thinking throughout the lifecycle encourages the development of well encapsulated classes — rather than considering encapsulation as an implemen-
In life, if you go to a shop as a customer and you ask for an example of a book publishing contract. Here is a similar product, the supplier has a responsibility to give what you asked. In one of Meyer’s papers (Meyer, 1989b) there is the concept that both receive benefits if the obligations are met. It’s not contractual obligations on both parties to the contract and benefits on both are displayed in the protruding box. The use of one or more of these services by the CUSTOMER O/C is indicated by the directed arrow. The base of the arrow is a filled circle indicating that the service is required to satisfy some internal requirement of the O/C CUSTOMER. If the service was required to satisfy the demands of the interface of the CUSTOMER O/C (essentially subcontracting) then the circle would be open and the service would appear in the interface of the CUSTOMER O/C.

The idea of contracting and subcontracting (Meyer, 1989b) is an important one. Although not yet fully developed, they are, although they correspond roughly in a diagonal sense in Figure 12. For example, with respect to the ‘deposit’ service, the obligation on the customer is to deposit an amount which is positive. If that obligation is met, then the benefit to the supplier is that the bank holds additional funds it can invest elsewhere, the benefit to the customer (client) is that the amount is credited and available for future use together with an expectation of future interest payments and the supplier can fulfill its obligation. If the customer attempts to deposit a negative or zero amount, thus breaking their ‘contractual obligation’, then the supplier (the bank account O/C) need take no action.

In a more mathematical example, you might design a class that contains a method to calculate a square root. The client has an obligation to send a message to that supplier which has arguments which are positive; to say ‘tell me the square root of 10’ or ‘tell me the square root of 3’. If the client breaks that obligation by saying ‘tell me the square root of minus 3’ then this regular square root method doesn’t have any obligation. The client has broken its own obligation, its part of the contract, to send it a positive value and so there is no obligation on the supplier to make any response.

You can code these responsibilities into a class in a declarative or procedural manner (but not in all OOPs at present). The contract (set of responsibilities) therefore says, in essence, that the client has an obligation to send a message to the supplier which has arguments which are positive; the supplier (client) is that the amount is credited and available for future use together with an expectation of future interest payments and the supplier can fulfill its obligation. If the client attempts to deposit a negative or zero amount, thus breaking their ‘contractual obligation’, then the supplier (the bank account O/C) need take no action.

In a more mathematical example, you might design a class that contains a method to calculate a square root. The client has an obligation to send a message to that supplier which has arguments which are positive; to say ‘tell me the square root of 10’ or ‘tell me the square root of 3’. If the client breaks that obligation by saying ‘tell me the square root of minus 3’ then this regular square root method doesn’t have any obligation. The client has broken its own obligation, its part of the contract, to send it a positive value and so there is no obligation on the supplier to make any response.

You can code these responsibilities into a class in a declarative or procedural manner (but not in all OOPs at present). The contract (set of responsibilities) therefore says, in essence, that the class functions are only available to you, as a client, if you fulfill certain obligations. There is therefore no obligation for the class to try and deal with situations for which it was not designed. A contract can be written using pre and post conditions and invariants within the class. The precondition states the obligation of the client and the postcondition the obligation of the supplier and are specified for each method separately. Sometimes there may be a constraint valid for all the methods in which case a class invariant may be a useful addition.

The idea of contracting and subcontracting (Meyer, 1988) is an important one. Although not yet fully deve-
Polymorphism is a very powerful concept to which I will give only an introduction since the subject becomes very complicated very rapidly. Polymorphism permits sending a message to different objects in the same inheritance hierarchy which results in different, yet appropriate, behaviour. So a message 'print' to a set of different graphical figures results in the use of different code. Furthermore, if graphical figures and textual documents belonged to the same inheritance hierarchy, the same message, 'print', could be sent to both graphical and textual documents and the appropriate printing mechanism could be invoked. This is actually a special type of polymorphism known as \textit{ad hoc} polymorphism or overloading (see eg Booch, 1991, p 103; Winder, 1991, p 131; Budd, 1991, p 187). Polymorphism also permits dynamic substitution of objects of different types. A definition states that when we wish to make an assignment \(a := b\), when \(a\) and \(b\) are complex types (ADTs) and are instantiations of the same ADT then of course there is no problem. However, it is also reasonable to make such an assignment when \(a\) is of type \(A\) and \(b\) is of different type \(B\) if and only if \(B\) is a descendant of \(A\). In other words, as long as \(B\) is somewhere in the inheritance hierarchy below \(A\), then the assignment can be realised. If \(A\) is a descendant of \(B\) it doesn't work (see eg Meyer, 1988, p 224). What does that mean? Here is an analogy (Meyer, 1988) that is perhaps easier to understand. I go to a restaurant and I ask for vegetables (class \(A\)) and the waiter brings me green vegetables (class \(B\)), then that's OK because \(B\) is a subset of \(A\) (green vegetables are, after all, vegetables). However, the converse doesn't work ie if I ask for green vegetables (\(B\)) and get vegetables (\(A\)) which might include carrots and cauliflower which are not green then I will reject this. Not OK. In a software context, that means that we don't have to know when we start the run, what type our objects have to be. We may send a message of \texttt{current\_balance}\(^7\) to a bank account \(O/C\) which may sometimes be of type \texttt{SAVINGS ACCOUNT} and at other times in the execution of the program be of type \texttt{CHEQUING ACCOUNT}. In both cases we may expect appropriate interpretation of our message of \texttt{current\_balance}, although the code used to implement it may well be very different. These are the basic ideas underlying polymorphism. For further details see eg Meyer, 1988; Blair et al., 1990; Budd, 1991; Meyer, 1991.

\subsection*{4.5 Other features}

In most OOPs polymorphism is strongly linked with dynamic or late binding (see also Winblad et al., 1990, p 37). The use of dynamic binding allows us to defer binding until run-time. Binding is the specification of the exact nature of the attribute: its name, type and storage location. Binding can be at various stages: it can be at language definition time, so \texttt{INTEGER} in FORTRAN is part of the language. It can be at compile time, which is early or static binding — and consequently doesn't permit the use of polymorphic calls. Sometimes the use of dynamic binding is mandatory; in other languages it is optional (eg in C++).
A *generic* class (eg Meyer, 1988) (also known as a parameterised type [eg Stroustrup, 1988]) is a special type of class in one which has one or more arguments of unspecified type. Using this feature of an OOPL allows, for instance, a generic array to store sometimes integers, sometimes reals and sometimes bank accounts.

I may wish to write classes which have a generic type associated with them. For example, I could write a class `ARRAY` (although I would actually just use the class from my class library and never have to write an array description again!). In procedural languages if I had an array in which I wished to store integers, I would have to state `ARRAY` to be of type `INTEGER`. If I wanted to use an array later in the program to store reals or, in object-oriented languages, bank accounts, I would have to declare a second array of this different type. In object-orientation I can have generic type of `ARRAY` in which I can store elements of any type. This is a single class to be coded which will have certain instantiations in integers. At other times, the same class may be instantiated to hold characters, or bank accounts etc. I can even have a mixed set of types, so long as they are all part of the same inheritance hierarchy. Those types of the element which are stored in the array here, say of type `BANK ACCOUNT`, are defined at run-time. Running the program, you sometimes instantiate one array as containing integers, another time bank accounts etc. At compile time, you don’t know what type of elements are going to be stored in this class array. For further discussions on genericity, see Meyer (1988).

5 SUMMARY

Although object-oriented computing and information systems are seen by some as the ‘flavour of the month’ or the ‘fad of the year’ (cf Page-Jones, 1991), the increasing involvement of major hardware and software vendors and the inception of the Object Management Group (eg Barber, 1991) suggest that the use of object-oriented techniques is likely to increase over the next few decades to become a new standard technique alongside structured development techniques. Increasingly, evidence is becoming available from large ‘commercial strength’ systems (eg McCullough and Deshler, 1990) to substantiate the claims of improved quality software being engineered through the use of object-oriented technology. Avoiding arguments as to whether object-oriented techniques are likely to replace procedural techniques or to co-exist with them, it has become obvious that all information scientists and computer technologists need to become conversant with at least the basic ideas of the object-oriented philosophy.

Here, I have presented those basic ideas without becoming embroiled in too much detail. Such details are readily obtainable in more specialised texts and papers.

ACKNOWLEDGEMENTS

I wish to thank Julian Edwards for commenting on an earlier draft of this manuscript.

REFERENCES


---

8Genericity is available in CLU, Eiffel and Ada but not currently in C++ (but cf Booch, 1991, p 118). In C++ it can be simulated using type conversions, or casts, a technique viewed by Meyer (1990) as defeating any attempt at maintaining a secure static typing system.


**BIOGRAPHICAL NOTE**

Brian Henderson-Sellers is Associate Professor in the School of Information Systems at the University of New South Wales. His current research interests include object-oriented systems development methodologies and notation; implementations of the object-oriented paradigm in the commercial environment; environmental decision support and simulation modelling. He is Convenor of the Object-Oriented Special Interest Group of the Australian Computer Society (NSW Branch).
1 ELECTRONIC FUNDS TRANSFER AT POINT OF SALE

Electronic Funds Transfer at Point of Sale (EFT/POS) is an application of information technology for which a very bright future had been predicted, but whose growth rate has been subdued. This paper provides a study of a particular approach to EFT/POS which has successfully addressed the factors which have held back progress. Although some aspects of the application may not be transportable due to particular cultural and institutional factors operating in Switzerland, both the technology and the principles are applicable in other countries.

Keywords and Phrases: EFT/POS, consumer, retailer, bank, financial institution, payment systems, credit card, debit card, smart card, chip-card, open systems.

principles can be reliably inferred from a single case is of course very limited. The circumstances are, however, similar to those which hold in other countries, including Australia. It is therefore suggested that careful consideration of this report may be of benefit to parties throughout the world who are currently contemplating, planning, designing, constructing, implementing, operating, modifying or abandoning EFT/POS systems.

2 RELEVANT ASPECTS OF THE SWISS ENVIRONMENT

Switzerland occupies an area of 40,000 square kilometres (about two-thirds the size of Tasmania), and half of that is steep alpine country. In that small space, however, it has a population of 6.3 million, about the same as New South Wales. It is in at least one sense the wealthiest country in the world, with a per capita Gross National Product about double that of Australia.

There is considerable diversity and decentralisation inherent in the country’s culture (four languages, with many dialects) and political system (23 cantonal administrations and a federal government with very limited powers). The deep-rooted scepticism about centralised power extends to the marketplace, where few industries are dominated by a single company. The marketplaces for financial services in general, and credit-card services in particular, are lively, competitive arenas, with a multiplicity of cards in use, issued by a multiplicity of banks and non-bank financial institutions.

There is considerable similarity between the retail sectors of Switzerland and of other advanced Western nations such as Australia. Supermarkets are similarly distinguishable from department stores and specialty stores, and the general design of, and procedure at, checkout counters is moderately consistent with those in many other countries. A considerable proportion of sales are in cash, a limited number are by cheque (more than in Australia, but far fewer than in the United Kingdom and France), and there has been a growing acceptance of credit card payment, initially using manual vouchers and ‘flick-flacks’.

The nature of the Swiss banking sector is fairly familiar to an Australian observer, with a small number of very large and long-established banks controlling the payments system, and thereby resisting incursions from the many smaller and newer banks and non-bank financial institutions. In general, banking services are a little less advanced than they are in Australia. In particular, most ATMs are now being connected on-line to account details in the host bank. On the other hand, the services are fairly well integrated, in that the ‘ec-Card’ (the ‘ec’ stands for ‘Euro-Cheque’) is issued by all of the major banks, and is accepted not only by all of their ATMs, but also by terminals at most petrol stations. A company owned jointly by the major banks, Telekurs AG, runs a single processing centre, which holds details of valid cards and monthly balances. This contrasts with the situation in Australia, where, even after an amount of rationalisation has taken place, there are still three separate ATM networks accepting, respectively, the cards of two of the four major banks and some other institutions; those of the other two majors and some other institutions; and those of a large number of non-bank financial institutions. The various Australian EFT/POS systems are now fully inter-connected, however.

There is a significant difference between the payments system in Switzerland and that in Australia. In common with most European countries, the dominant method of payment by Swiss consumers has been and continues to be via the national PTT’s ‘giro’ system. Cheques have only come into common use during the last decade, and appear unlikely to ever represent a significant percentage of transactions.

The transition from cash-registers to more complex point-of-sale terminals has been underway in Switzerland for some years. To date, the primary developments have been for large corporations to integrate their point-of-sale terminals more effectively into their own inventory and purchasing systems. In some cases, proprietary EFT/POS functionality has been incorporated in point-of-sale terminals, enabling the retailer’s own card to be used as an alternative to cash. Until the implementation of the system reported on in this case study, there had been no installation of an open system capable of accepting a variety of cards.

The Swiss PTT (roughly equivalent to Telecom Australia after its absorption of the Overseas Telecommunications Commission, OTC) provides sophisticated telecommunications services. It has been far in advance of the PTT of its large neighbour, Germany, about the equivalent of the (very differently structured) United States system, and not markedly inferior to the world leader, Telecom France. The Swiss PTT is well-advanced with its introduction of chip cards, with pilots completed, and some half-million now issued. To date their principal purpose has been to carry a complete audit trail of transactions and the current balance on the card-holder’s account, capable of being displayed on a device which is not attached to the network. A multi-function chip card is being piloted (PTT 1988, 1990).

This report reviews a particular EFT/POS application, which is being progressively installed throughout a large chain of Swiss supermarkets, using products developed by a Swiss company. The interests of the retailer, its customers, and the primary supplier are considered. The report concerns itself with the period between 1986 and 1991, and is based on interviews conducted in January and June 1991 with senior executives of the two main companies involved, a demonstration of the equipment and procedures, observation and use of the system in supermarkets,
Because the system was launched by a retailer rather than a bank, it is of far more than merely local or European interest. Although the details of implementation are specific to the Swiss environment, it is possible to infer some general principles. In addition, the equipment and software are directly applicable in other countries which have adopted international telecommunications standards.

3 MOTIVATIONS

3.1 The Retailer's Perspective

Migros is a large Swiss conglomerate, comprising a dozen regional cooperatives operating a number of services, including supermarkets, retail stores, service stations, travel agencies and a bank. A joint Head Office operates out of Zurich. The chain of over 500 supermarkets is the largest in the country, with 1990 turnover of about 10,000 million Swiss Francs (about the same in Australian dollars). Migros' own Bank is small in comparison to the four giant institutions, but is nonetheless significant in terms of the value of turnover and deposits, and the number of customers and cards on issue.

Migros has a very strong commitment to community service, a philosophy which is expressed in its Constitution, and embedded in its managerial and operational culture. The Migros cooperative movement has had a long and strong aversion to credit, and to credit cards, and its EFT/POS services have therefore been based on debit-cards, which access the card-holder's own funds, rather than borrowing against a line of revolving credit.

In the mid-1980s, the Finance Director of the Migros Group of Cooperatives, Hr J. Kaufmann, was concerned that EFT/POS systems would be implemented by the banks, under conditions and charging structures suitable to themselves rather than the retailers and consumers. This has been the case in many countries, particularly in the country's neighbour, France. The Group therefore decided in 1986 to develop its own EFT/POS system which would be 'open' in the sense that it would accept, and process on-line, cards issued by many different organisations. Migros and other retailers formed an association in 1988, which has successfully precluded the banks from controlling EFT/POS in Switzerland, and ensured that the openness concept is intrinsic to the Swiss EFT/POS system.

Participants whose interests needed to be taken into account were retail merchants, banks, the PTT, and manufacturers and suppliers of hardware and software. Most critical, however, were the customers (and, in Migros' case, members). Exhibit 1 shows Migros' interpretation of their most critical requirements (translated from Pfister 1988).

Exhibit 2 summarises Migros' view of the most important cost and benefit factors (translated from Pfister 1988).

For both banks and retailers, cash transactions are relatively expensive (because of the high cost of labour-intensive cash receipt and dispensing), and so too are cheque transactions, which require physical handling and capture, and whose full costs are of the order of 2-3 Francs.
per cheque. Since the cost of EFT/POS transactions (assuming moderately high volumes) is of the order of 20-80 cents each, the costs of banking could be reduced by replacing cash and cheque transactions with electronic ones. The extent to which these savings would accrue to the bank or the retailer would depend on negotiations. In addition to reductions in cash-handling costs, the lower opportunity costs of cash-holdings would be significant.

There were potential benefits for Migros' customers/members also. One source was lower prices resulting from the lowered cost-structures of Migros and the financial institutions. More directly, there was a possibility of time-savings at checkout-counters. For example, a study in Zurich showed that about half of the time spent at checkout-counters was involved with payment rather than the handling of goods, and that time and money might be saved. Where the customer pays by cheque (which is common in countries with cheque guarantee cards, such as the United Kingdom and France), payment requires some two minutes after the cost of the goods has been totalled. A reduction to, say, 30 seconds, would therefore result in measurable benefits such as fewer checkout positions, staff salary savings and/or more satisfied customers, and more space for product display (Hinnen et al. 1987). These studies are not conclusive, however. Payment in cash is a relatively quick procedure typically involving about only 15 seconds after the cost of the goods has been totalled. Hence, where electronic means means replace cash payments, EFT/POS offers less significant time-based financial savings or other improvements. Other advantages to consumers are the convenience of cashless shopping, the decreased risk of loss of cash, and the possibility of withdrawing cash at the checkout till.

Other important changes are under way in point of sales operations, including the scanning of machine-readable product-numbers and the auto-weighing and charging of bulk and fresh goods, both of which relieve the checkout assistant of keying some or all of the prices or goods identification codes, and may in due course lead to self-service checkout, with resulting labour-savings. Migros chose to deal with those technologies in a separate (though related) project, and this case study accordingly leaves those developments to one side.

3.2 From the Primary Supplier's Perspective

Ascom Autelca AG manufactures a variety of equipment, including coin-operated telephone and ticket vending machines, ATMs and EFT/POS terminals. The company operates out of Bern, the Swiss capital. With its experience in ATM networks for the banks and the PTT, and petrol station EFT-terminals, the company anticipated that card-issuers would not be able to agree common technical requirements for EFT-terminals. It concluded that competitive advantage would accrue to a supplier which could implement, using a single family of terminals, applications which satisfied each card-issuer's specifications, read cards which used different technologies (in particular, magnetic stripes and chip-cards), and provided a range of alternative security levels.

Ascom Autelca's motivations were therefore to develop a new line of business leveraging on their existing strengths, and thereby to enter a new market which provided significant growth prospects. The company had a strong preference to develop a family of products which would satisfy a real consumer need, rather than adopting a technology-driven approach. In this way, the company reasoned that it would maintain long-term market-share and volume, gain repeat business, and thereby have a longer period in which it could recover its substantial initial investment. During the company's early discussions with Migros, in the mid-1980s, it was apparent that this desire for a genuine, long-term solution to the needs of the various parties, was common ground.
4.2 Supplier Strategy

Ascom Autelca is not a cash-register manufacturer, and did not intend entering into competition with such companies. The company wished to address, at least in the first instance, the large retailers, who have a significant number of suppliers. Together the models cover more than half of the current market in Switzerland. During 1989 investigation into a case of ‘pillow talk’ by the then Attorney-General to her financier husband, it accidentally became public that the Federal Police had for nearly 40 years maintained secret surveillance of some 140,000 Swiss citizens and 580,000 aliens. This has resulted in the traditional Swiss scepticism about large organisations (public or private) being exacerbated.

Finally, Ascom Autelca did not perceive EFT/POS in major retail stores as a phenomenon independent of other areas of development. EDI is growing quickly, and public access to a variety of services (such as directories, public transport timetables and reservations, home banking and tele-marketing/tele-shopping) is emerging. The Swiss PTT is running a number of community-based pilot projects, including an ‘electronic cash’ or ‘value-card’ system, in the city of Biel (PTT 1988, 1990). Ascom armed for a conception which would enable it to capitalise on its investment in EFT/POS technology by using equipment from the same family to support such other innovative applications as appeared likely to gain in popularity.

5 THE CARDOMAT EFT/POS SYSTEM

The planning phases of the Cardomat/Migros project took place in the period 1985-86. Migros gave the go-ahead for implementation in May 1986, and the first site ‘went live’ in April 1987, initially supporting only Migros’ own card. In June 1987 the PTT agreed that its cards could be used on the system, and this service went live in February 1988. Agreement was reached with the banks in June 1989 for the ec-Card also to be accepted in open EFT-terminals. The specifications for this were published in December 1989, and implemented in June 1990. The remainder of this paper concentrates on the scheme as it has been piloted in several of the regional cooperatives, and is in the course of implementation in the remainder, throughout Switzerland, during 1991-92.

5.1 Overview

The hardware, software and procedures were developed by Ascom Autelca AG, in close association with Migros. The generic product is called EFTOMAT, and the initial version implemented with and for Migros is called CARDOMAT. To avoid the amount of information becoming unmanageable, this case study has been restricted to the Migros implementation. Exhibit 3 provides a simplified overview of the system as a whole, comprising point-of-sale equipment, concentrators, network and host machines.

Ascom Autelca has delivered its software in a form compatible with cash registers from a number of different suppliers. Together the models cover more than half of the current market in Switzerland, and a significant propor-
Exhibit 3: Cardomat/Migros Overview (based on Migros Diagram of December 1990)

<table>
<thead>
<tr>
<th>MIGROS SUPERMARKET</th>
<th>CUSTOMER AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office</td>
<td>Retailer</td>
</tr>
<tr>
<td>Cardomat Concentrator</td>
<td></td>
</tr>
<tr>
<td>Migros-Bank Processor</td>
<td>PTT Processor</td>
</tr>
<tr>
<td>Telekurs Processor</td>
<td></td>
</tr>
<tr>
<td>Host Bank Processor</td>
<td></td>
</tr>
</tbody>
</table>

The national X.25 network, Telepac, operated by the Swiss PTT, is the main message-carrier, although the public switched network can also be used, depending on the type of terminal.

The following sections present the point-of-sale equipment and concentrator design, the cards supported and services provided, and the operating procedures from both the customer's viewpoint and the technical perspective. A final sub-section discusses security, integrity and reliability aspects.

### 5.2 Point-of-Sale Equipment and Concentrator Design

There are two alternative terminal designs, one mainstream, the other transitional. The transitional arrangement is for sites which have recently installed new cash registers which are not programmable, or which are not able to support an additional interface. Because this component raises no substantial issues, it is not further reported on here.

The **Integrated Cardomat Terminal** comprises two physically separate elements:
- an electronic cash register, supplied by a cash register supplier, but incorporating software and interfacing defined by Ascom Autelca; and
- a customer module supplied by Ascom Autelca.

The **Electronic Cash-Register** (ECR) comprises:
- a device for capturing price or goods identity data (i.e., keyboard or scanner);
- a display screen for the checkout assistant;
- a display screen for the customer;
- a cash drawer;
- a printer, for customer receipts;
- an audit trail mechanism (e.g., separate printer and/or cassette or diskette);
- a programmable processor;
- a program-loading facility (typically one or more slots for ROM-cards).

The ECR must be fitted with software and an interface which implements Ascom Autelca's specifications. The functions performed are:
- acquisition of the amount of the transaction from the ECR's internal storage;
- communication with the host (via the Customer Module and Concentrator);
- receipt of a return code; and
- passing of the authorisation back to the ECR.

Naturally the implementation involves close cooperation between Ascom Autelca and the cash register supplier. In accordance with Migros' long-standing policy, several suppliers are approved, including NCR (a long-standing supplier of point of sale equipment to Migros), ICL (which has not previously been a supplier, but has long experience in the retail industry), Sweda and IBM. Under Migros' Constitution, each regional cooperative is free to make its own decision as to which supplier and model it selects.

The **Cardomat Customer Module** is a separately housed device, manufactured and supplied by Ascom Autelca (see Exhibit 4). It comprises:
- a card-swipe, for ISO 2/3 cards and chip-cards;
- a numeric keyboard;
- six special keys; and
- a small display screen.

The keyboard and display are recessed. The keyboard is set at 20° above the horizontal. The 20-character display is set above the keyboard and at a higher angle. The numeric keys have the conventional alphabetic equivalents printed on them. The six special keys are colour-coded. The device may be table-top, stand- or wall-mounted, with the card-reader either vertical or horizontal.

Each ECR is connected to its Customer Module, and each Customer Module to a purpose-designed **Cardomat Concentrator** which connects up to 93 devices to Telepac, and hence to the card-issuers' machines. The concentrator comprises a processor, modules handling communications with both the terminal modules and Telepac, storage (a
CARDOMAT/MIGROS

Exhibit 4: Cardomat Customer Module.

hard disk), and a printer. In addition to its primary, real-time functions, the concentrator maintains an audit trail (which includes the card-id and transaction amount, but under no circumstances the customer account-balance). The journal and traffic statistics can be printed and/or uploaded to the merchant’s own processor, to enable centralised network management and problem-investigation.

The Cardomat Customer Enquiry Terminal comprises a stand-alone Cardomat Customer Module, with software which provides a display of the account balance, or the amount remaining available for use during the current day or month (which of these applies, depends on which card the customer uses). This class of terminal is installed in the entrance area to supermarkets, to allow customers to check their balances, particularly before and after making purchases.

The Infotel Customer Enquiry Terminal provides a service exclusively for holders of PTT Postomat-Plus cards. Each such chip-card contains an audit trail of transactions undertaken using it, and this terminal is a stand-alone unit which reads the chip and displays the transaction data it contains. Infotel terminals are owned by the Swiss PTT, but are relevant to this case study partly because they are manufactured by Ascom Autelca, but particularly because they are installed at Migros supermarkets at the same time as Cardomat, as a service to customers, and a means of piloting chip-card services for the public.

5.3 Cards and Services Supported

Three cards are currently operational under the Cardomat/Migros system:

— Migros’ own M-Card, a conventional magnetic-stripe card, issued gratis to account-holders by the Migros subsidiary Migros Bank;
— the PTT’s Postomat-Plus Card, a chip-card issued gratis to account-holders by the postal service; and
— the ec-card, issued by the major Swiss banks to account-holders against an annual fee. It is a conventional magnetic-stripe card.

The terminals are on-line to the Migros Bank’s system, enabling on-line verification. The PTT database, on the other hand, is only updated periodically, and the ec-Card service is on-line only to the Banks’ front-end processor run by Telekurs AG, not to the Banks themselves, and the Telekurs database is only updated nightly. The services available therefore vary:

— purchases up to the level of account-balance (M-Card), Fr.3000 per month (Postomat-Plus Card), or Fr.2000 per day (ec-Card);
— cash withdrawals up to Fr.1000 per transaction (M-Card), or Fr.300 per transaction (Postomat-Plus Card).

At present the Banks do not permit cash withdrawal against an ec-Card; and
— display of the current account balance (M-Card), the purchase-balance remaining for the month (Postomat-Plus Card), or the card-limit remaining for the day (ec-Card).

The M-Card service has been available since the first trials of Cardomat in 1987. Postomat-Plus cards have been supported since 1988, and are important to Migros because of the very large number of people who are used to using PTT services, and the very large card-base. Because it is a chip-card, and contains a complete audit trail, it is also possible to display the transactions which have been undertaken using the card; this service is only available on Infotel Customer Enquiry Terminals, however.

The ec-Card service was added in 1990, and involves delicate ‘feeling-out’ of the bounds of collaboration within a competitive environment. This card can also be used for a number of off-line services, including as a cheque-guarantee card, at 2,500 Bancomat ATMs owned by the banks, and at off-line terminals at about 2,000 petrol stations.

It has been an important aspect of Ascom Autelca’s strategy to ensure that, in principle, any card and a wide range of services, can be supported within the existing architecture, merely by implementing an appropriate version of the software in the Customer Module and/or Concentrator.

5.4 Cardomat Operating Procedure — Customer Viewpoint

To pay for goods using Cardomat, the customer swipes the
card and keys the PIN. A series of instructions and messages is shown on the small display. Except for the final step, the customer may undertake these actions independently of the checkout assistant’s registration of the customer’s goods. The response is displayed within a few seconds. A purchase may be paid for wholly by card, wholly in cash, or partly in cash and partly by card, and the customer can tell the checkout assistant which option he or she wishes to use at any time until the registration of the goods has been completed. Depending on which card-type the customer uses there are minor differences in the procedure, such as when the card can be removed from the cradle, and whether it is necessary to strike the ‘OK’ key to complete an action.

Cash withdrawal (available in the cases of M-Card and Postomat-Plus card, but not in the case of the ec-Card), is not included as part of a purchase, but is handled as a separate transaction. The amount of cash which may be withdrawn is limited, but the frequency with which withdrawals may be undertaken is not.

The Cardomat Customer Module is designed to deal appropriately with all conceivable errors and idiosyncrasies in the customer’s actions. For example, one circumstance which was inadequately handled by early versions of the software was repeated swiping of the card (eg due to slow response on the display panel). The messages appear in the language appropriate to the language-code on the card (German, French, Italian or English). A part-completed transaction can be annulled by the customer at any time until the message is sent to the host.

The printed voucher contains full details of the transaction, including date, time, shop and location, the nature of the transaction (purchase or cash withdrawal), the prices of the individual items (but until such time as bar-code scanning is introduced, not the item description), the total, the amount paid in cash and by EFT/POS, the card number and a voucher number (EFT-number). Transactions are printed, including varying degrees of detail, on the periodic account statement provided by the card-issuer to the card-holder.

5.5 Cardomat Operating Procedure — Technical Perspective

The manner in which a Cardomat transaction is processed is a function of a number of factors, particularly the cost-structure set by the PTT for Telepac packets, the design of the card-issuer’s processing system, and the presence or absence of a processor on the card itself.

The verification functions which need to be performed are:

— **PIN verification**: that the PIN is correct for that card;
— **account verification**: that the account with which the card is associated is available for the transaction (in particular it must exist, must not be barred, and must be empowered to conduct the particular class of transaction). Note that the relationship between cards and accounts is not 1-to-1 but many-to-1, ie more than one card may be associated with a single account, typically one for each spouse; and
— **amount verification**: that the amount of the transaction is within the approved limit.

The **processing functions** which need to be performed are:

— debit to the card-holder’s account with the card-issuer;
— credit to the merchant’s account with the card-issuer or to the card-issuer’s financial institution, as appropriate to the circumstances;
— provision of a receipt to the customer; and
— provision of a voucher to the merchant.

In addition, a set of **audit trails** must be maintained, which is sufficient to resolve ambiguities and disputes and provide evidence to the courts, but is not unduly expensive given the low frequency and financial significance of such ambiguities and disputes.

The manner in which these functions are performed is different in each case, because of the different configurations of the cards, accounts and services involved. In the case of **Migros’ own M-Card**, the procedure is as follows:

— when the customer has swiped the card, keyed the PIN and quoted with the ‘OK’ key, and the checkout assistant has approved the transaction amount, a transaction is passed via concentrator and Telepac to Migros Bank’s processor;
— all verification functions are performed in the Migros Bank processor. Because the Migros Bank has its account balances on-line, the transaction amount is tested against the funds in the account (Migros does not provide consumer credit, and hence the test is against the actual balance in the account); and
— if all verification tests are successful, the processing functions are performed, and an affirmative return code is sent via Telepac and the concentrator to the Cardomat terminal, causing a receipt to be printed. If, on the other hand, the transaction is rejected, the return code is negative, and a delicately phrased message is displayed to the checkout assistant.

In the case of **the PTT Postomat-Plus card**, the procedure is similar. The most significant difference is that the PIN verification function is performed in the chip on the card itself, by comparing the PIN keyed against that stored in the chip. Another difference is that the PTT authorises a maximum amount in any calendar month, and the amount verification is therefore against an approved maximum amount rather than against an account balance. Because
Postomat is a 'smart card', further possibilities exist. However the card does not at present contain data such as account balance or card and account barring. Such data could be down-line updated from the host, but with a delay factor of one business day, because the host database is currently updated overnight in batch mode. Until the host is updated on-line and accessed from Cardomat terminals on-line, increased sophistication in Postomat-Plus card processing seems unlikely.

The processing of the ec-Card differs in the following significant respects from the other two card-types:
— as soon as the card is swiped, the card-number is transmitted from the Cardomat Customer Module via the concentrator across Telepac to the Telekurs processor;
— the card and account verification functions are performed on the Telekurs host. If appropriate, a rejection is sent to the Cardomat Customer Module. If, however, the transaction is conditionally approved, a 'PIN offset' and the maximum authorised amount (based not on account balance, but rather a daily limit) are transmitted to the terminal;
— the Cardomat Customer Module performs the amount verification process;
— PIN verification is performed in another way entirely. The PIN off-set is received in the Cardomat Customer Module from the Telekurs host via Telepac and the concentrator. The PIN-pad contains a security-module, supplied by a Belgian company (and whose internal specifications are unknown to Ascom Autelca), which verifies the PIN keyed against that expected on the basis of the PIN offset; and
— if all verifications are successful, a second transaction (comprising essentially card number and amount) is transmitted from terminal to host, and a receipt printed at the terminal. When acknowledgement is received from the host by the Cardomat Customer Module, the transaction is completed.

The ec-Card solution, although in some respects more complex and therefore more expensive to implement and more error-prone, enables a shorter delay from approval of the transaction at the point-of-sale until the receipt commences printing (reported by Ascom Autelca to be about two seconds compared with about six seconds).

From the perspective of this case study, and EFT/POS transaction processing in general, the diversity of solutions appears to be excessive. The reasons are partly historical (modifying the various host systems to optimise Cardomat/Migros' simplicity would have been slow, error-prone, expensive and politically difficult), partly based on political and competitive factors, partly based on philosophy (Migros Bank was not prepared to transmit account balance to a terminal on the premises of another organisation), partly based on differing assessments of the future directions of the PTT's charging structure, and partly based on differing assessments of future host response times (eg processor and storage capacities, and software and database management system characteristics).

5.6 Security, Integrity and Reliability Aspects
Data security and integrity are important considerations (see, for example, Schmitz 1989, Weber 1989 and Clarke 1990). From the viewpoint of the banks and merchants, it was necessary to ensure that funds transfer was accurate, reliable and resistant to interception and unauthorised modification. This was in large measure achieved through the use of the Telepac network, PINs, and in the case of Migros' own card, on-line account balance checking.

A transaction may not be approved due to difficulties with the card. The major circumstances in which this arises are:
— multiple unsuccessful PIN entries;
— stopped cards, in particular those that have been reported missing, and those which have been cancelled by the card-issuer, eg due to insufficient funds;
— out-of-date cards, ie whose period of validity has expired; and
— early cards, ie whose period of validity has not yet commenced.

In all of these cases, the offer of payment by card is simply declined by the checkout assistant, and no further action is taken — Migros has refused to perform any security functions such as the capture of stopped cards.

From the viewpoint of the reliability of the service, the terminals include an emergency electricity supply which enables an appropriate completion or suspension of transactions which are in process at the time of a power failure. During periods in which the system is unavailable, a fallback system involving paper vouchers is used. For Migros cards and PTT cards, this uses an 'emergency voucher'. This is accepted by the card-issuer at no risk to the retailer for an amount of no more than SF300. Although this is in principle a very insecure mechanism, it is relied upon sufficiently infrequently and unpredictably, and there has been a sufficiently low incidence of fraud, that it is retained as a means of customer convenience.

In the case of the ec-Card, the only fallback is the use of a Euro-Cheque (which is a payment instrument issued to bank customers in blocks of 10 looseleaf sheets, rather than in bound booklets of 30 or more sheets as is normal in the United Kingdom and Australia).

6 PRESENT AND FUTURE

6.1 Implementation Experience

From a marketing perspective, it was critical that the public quickly come to terms with the system and its advantages for them. Brochures and advertising placards concentrated on pictures rather than text, on the extra interest which could be earned by keeping cash in a bank account rather than in a wallet or purse, and on speedier service at the checkout counter.
Although deposit rates in Switzerland are extremely low (1.5-4% pa cf 7-11% pa in Australia), the volume of cash is very high, due to the absence of a widely-used chequing system, and a standard of living about double that of Australia. On the assumption that the amount of cash carried by Swiss people is of the order of 1-5000 Swiss Francs, conversions to cashless operation would offer an actual interest-income difference of 30-200 Swiss Francs pa. A further promotional measure was the offer, during the implementation phase, of a bonus of 0.50 Swiss Francs for each cashless purchase transaction.

The level of use during pilot implementation in 1986-87 was 50-100 transactions per day for purchases and cash withdrawals. The proportion of all purchases undertaken using Cardomat has been as high as 12% (in Winterthur, a medium-sized, German-speaking industrial city). The size of cashless transactions was on average higher than for cash transactions. There was a further 25-50 transactions per day for display of account balance alone (ie at Cardomat Customer Enquiry Terminals), which suggested firstly that many consumers were taking advantage of the ability to check the accuracy of transaction processing, and secondly that there may be a strong desire to avoid embarrassment at the checkout counter by having too little in the account to cover the purchase. To date, on the other hand, use of Infotel Customer Enquiry Terminals has been very low. On the basis of surveys conducted to date, the cash withdrawal facility was regarded as a significant service, and the majority of checkout assistants regarded the Cardomat system as simple to use.

The self-service principle in relation to card and PIN has proven satisfactory in practice, and an essential step in paralleling the activities of registration and payment, and hence reducing, or at least maintaining, the speed of service at the point of sale. The time spent at the checkout counter, however, has proven to be scarcely any different from that for cash-based transactions, which is in contrast to expectations, and to the advertising brochures.

From a cost/benefit perspective, the merchant is faced with additional equipment investment of several thousand Swiss Francs. After the initial staff training costs, some ongoing refresher training and training of new staff must also be taken into account. Reconciliation and dispute-resolution costs are currently higher, although this would appear likely to be a transitional factor. On the positive side, funds reach Migros' own bank accounts on the same day as the transaction instead of one day later, and bank charges for cash-handling of about one Swiss Franc for every 1000 in cash turnover are able to be saved. Other advantages which have been realised are:

- a system open to all cards in principle, and many cards in practice;
- retailer choice of cards which are to be accepted;
- no commission paid by the retailer on debit-card transactions;
- decreased commission paid by the retailer on credit-card transactions; and
- communication costs paid by the card-issuer.

6.2 Future Developments
Cardomat is being progressively installed in all large and medium-sized Migros supermarkets, subject to the decisions of the individual regional cooperatives. It is being considered for Migros specialist retail outlets, in insurance (Secura), bookshops (ex libris), travel agencies (Hotelplan) and service stations (Migrol).

Ascom Autelca is actively selling the generic product, EFTOMAT, to other merchants, particularly to the larger specialty shops, and to stores operated by Migros' major supermarket competitor, Coop. In its present form, the system is less applicable to small, single-site retailers.

In principle, EFTOMAT is capable of connection to a wide range of card-issuing organisations, including credit-card companies, although in Migros' case, the cooperative's philosophy and Constitution precludes the acceptance of credit-cards. Coordination with the parallel EFT/POS developments of other large retail groups is being investigated. The Swiss subsidiaries of credit-card operators (importantly American Express, Diners Club, EuroCard and Visa) have published a specification to accept their cards at on-line terminals. (In Australia, the majority of transactions entered into EFT/POS terminals to date have actually been such credit-card rather than debit-card transactions). Cardomat is designed in such a way that the difficulties of accommodating these other cards is not great.

Petrol cards are issued by the major oil-companies such as Shell, BP and Esso, and also by the Migros subsidiary, Migrol. For some years now, most service stations have been equipped with terminals providing first-generation, off-line EFT/POS capabilities (integrated with unattended, cash-payment, self-service functionality), and it may therefore be some years before the opportunity arises for EFTOMAT to address this market.

There are also a number of important 'house' charge-cards in use in Switzerland, such as those of the major department stores, Globus, Jelmoli and Loeb, and it is technically possible for these to be accommodated by EFTOMAT. Jelmoli is of particular interest, because its house-card was first issued (on metal) in the 1930s, and it has a large, active and affluent card-base, perhaps exceeding 250,000. As with service stations, however, any expansion of EFTOMAT into this sector would have to be timed to coincide with the renewal of point-of-sale equipment.

It is possible that the banks may approve cash withdrawal, and provide on-line connection to their accounts, and hence improved verification of transactions and removal of arbitrary limits on card use. They may also perceive it to be to their advantage to offer higher transaction-subsidies.
to Migros, in order to encourage reductions in expensive cash, cheque and credit-card transactions.

7 CONCLUSIONS

This case study has described Ascom Autelca's EFTOMAT EFT/POS product, with particular reference to its first implementation, under the name CARDOMAT, in Migros' supermarkets. The system's technical features have been outlined, in the context of its economic, commercial and social purpose. The system resulted from synergy between Migros' vision of an open EFT/POS system, dominated by retailers rather than card-issuers, and the experience, orientation, strategy and capabilities of Ascom Autelca.

The gains from EFT/POS appear to be potentially large enough that each of the various parties (retailers, financial institutions, consumers and suppliers) can all win. The distribution of benefits among them, however, must be such that all parties have sufficient incentive to actively participate. Effective implementation is therefore contingent upon the critical success factors being identified and addressed at an early stage in the development of the system. Two clusters of important considerations arise from this case study. For further discussion of some of these points in the Australian context, see Clarke and Walters (1989) and Walters (1989) and (1991).

The first cluster of conclusions relates to system architecture:

- 'business openness' is vital — all EFT/POS terminals need to accept all cards;
- 'architectural openness' is vital.

Either:

- the infrastructure should be common; or
- if several networks exist, then interoperability must be achieved; and
- 'technical openness' is needed, to underpin the first two requirements. By this is meant commitment to international telecommunications standards.

The other cluster of conclusions relates to consumer acceptance:

- successful EFT/POS system design demands a strong affinity with the retail point-of-sale environment, and more generally with consumers;
- consumer convenience is vital, in particular through:
  - support for a wide range of cards;
  - speed of service at the checkout counter;
  - availability of a cash withdrawal service; and
  - provision of a facility to check funds availability;
- the consumer must be active in the payments process, through direct interaction with a customer terminal which accepts the card, displays instructions and information, and enables keying of the PIN;
- privacy-sensitive design is important, including shielded keying of the PIN, display of account-balance on the customer terminal only, and non-retention of personal data; and
- consumer education is vital (in particular, promotional material needs to stress the system's simplicity, convenience and economic benefits), and so is consumer encouragement (e.g. through a rebate on card-based transactions).

The extent to which these conclusions are applicable in contexts other than Migros is not addressed in this paper. It is apparent, however, that considerable similarities exist between the environment in Switzerland and that in many other advanced western nations. This case study may therefore contain important lessons for EFT/POS developers in other countries such as Australia.

ACKNOWLEDGEMENTS

This case study was possible because of the efforts of Hr Pierre-Andre Steim, Viziedirektor and Leiter, Planung und Organisation, Migros-Genossenschafts-Bund, Zurich, and Hr Eduard Derksen, Leiter Marketing, EFT-Systeme, Ascom Autelca AG, Bern. The comments of Michael Walters, of Retail Response Pty Ltd, Sydney, were also very helpful.

The paper has been reviewed by both companies for factual accuracy and reasonableness of the evaluative comments, and the permission of the companies to publish this material is gratefully acknowledged. Naturally the responsibility for errors, omissions, evaluations and opinions remains entirely with the author.

REFERENCES

**BIOGRAPHICAL NOTE**

Roger Clarke spent 17 years in professional, managerial and consulting work in Sydney, London and Zurich, prior to taking up the post of Reader in Information Systems at the Australian National University in 1984. His interests are in application software technology and its management, and organisational, economic, legal and social aspects of information technology.

He was employed by a Zürich-based software house in 1979-82 and undertook a Visiting Professorship at the University of Bern in 1987-88. He has no commercial association with any of the companies involved in this case study.

---

**International Conference on Theorem Provers in Circuit Design: Theory, Practice and Experience**

22-24 June 1992

Nijmegen, The Netherlands

Sponsored by IFIP TC10/WG 10.2 and the Dutch National Facility for Informatics (NFI)

**Focus and Objectives**

Formal methods are increasingly seen as important in the design of digital systems. The use of these techniques in practice is often regarded as being strongly dependent on the support of appropriate mechanised theorem proving tools. The purpose of this conference is to provide a forum for discussing the role of theorem provers in the design of digital systems. The objective is to cover all relevant aspects of work in the field, including original research as well as case studies and other practical experiments with new or established tools.

The primary focus will be on the ways in which formal methods are supported by theorem proving tools, rather than on the theoretical foundations of formalisms and design methods. The topics of interest include the philosophy behind such tools, their design and development, their evolution, and their evaluation through use. Of equal importance is the migration path of a theorem proving tool and the associated technology into current digital engineering practice.

The intended audience includes workers in the field of hardware verification as well as practising digital designers.

**Tutorials**

A Tutorial Chair has been established to ensure that a wide range of systems are represented and to underline the importance that is placed on the matter. A digest of the tutorial presentations will be included in the proceedings.

**Programme and Proceedings**

The conference programme will start with a day of tutorials and demonstrations, followed by two days of presentations by contributing authors. The programme will also include invited lectures by three prominent researchers in the field of machine-assisted verification. The invited speakers are Mike Gordon, University of Cambridge; Warren Hunt, Computational Logic Inc.; Dave Musser, Rensselaer Polytechnic Inst.

The conference proceedings will be published by North-Holland before the conference.

**Conference Organisers**

General Chair: Raymond Boute, University of Nijmegen.

Programme Chair: Victoria Stavridou, University of London.

Tutorials Chair: Tom Melham, University of Cambridge.

Local Arrangements Chair: Huub van Thienen, Harriet Reker, University of Nijmegen.

---

Department of Computer Science

Computer and Communications Systems Group

PO Box 9010 NL 6500 GL Nijmegen

Email: germaine@cs.kun.nl

This book is about using serial communications under UNIX, in particular under SCO UNIX and XENIX systems. Although the details of how serial communications work may differ in other versions from the ones discussed in this book, the general principles will still apply. The introduction indicates that it is intended to accompany a course on UNIX communications. The style is appropriate for that purpose.

The book gives some background by talking about serial devices and how they work and then describes the use of several serial communications programs available under UNIX. The latter topic is split into divisions reflecting different categories of users and their interests: general users, systems administrators, and programmers. The text is supported by tables, explanatory pictures and examples.

The introductory section starts with the hardware and, moving up through low level protocols, explains the terms and basic concepts underlying serial communications.

The general user is told how to use mail, transfer files and connect to remote machines. The systems administrator is told how UUCP works, how to configure it, and where to start with troubleshooting. The programmer is told how to access serial devices through UNIX system calls and is given a number of tips and techniques for interaction.

The appendices briefly describe some common pin connections, and the XMODEM, YMODEM, and Kermit file transfer protocols.

I had the following minor gripe: the 'Useful Command Tables' advertised on the front only cover the use of the mail program, which is probably the area of maximum variation between communications programs under UNIX. Because many different mailers are available under UNIX, the one covered here may not be useful.

True mastery of any subject cannot be obtained without practice, but for the beginner, understanding the contents of this book will help to provide the basis of knowledge upon which mastery can be built. The coverage seems fairly extensive although not exceedingly deep — but deep enough for the purpose, I think. I am not familiar with the specific systems mentioned, but the general information seems accurate. The layout is clear and easy to follow.

This book will be of particular interest to those who have no prior knowledge of serial communications or who find that the standard UNIX manual does not contain sufficient background.

L. Chubb
Software Pty. Ltd.


Every office worker who has laboured long and hard to conquer the intricacies of WP (word processing) or DTP (desk top publishing) packages and then been disappointed to discover that the resultant documents are dull, unattractive or ineffective, or all three; will find this book welcome and useful.

The book takes the reader beyond the instruction manuals which describe how to select fonts, format headings, layout pages etc., and advises on which fonts to select, what layout styles etc. to choose in order to enhance the readability and effectiveness of documents. A number of tables and diagrams giving examples and rules-of-thumb for good design are included for those who want a quick, recipe-book approach to improving their work. For those who want a deeper understanding of the principles involved, the text provides a more solid basis. The chapters are clearly laid out into concise subject areas to facilitate speedy referencing at whichever level the reader wishes to approach the book.

The text is easy to read, with technical terms explained adequately. Short excursions into the history of printing and typesetting evolution make for interesting reading and may help the reader better understand some of the more esoteric components of WP and DTP software and manuals. This background material, combined with some of the later chapters on the printing process, will be useful when liaising with printers in the production process.

A minor shortcoming is the lack of a bibliography. The author makes a number of general references to research on the suitability of fonts and styles, but omits any specific references which may allow a reader to follow these lines of research. In a similar vein, there is no synopsis of the author's background. There is a useful glossary (although with one or two errors) and an easy to read index.

The book is not specific to any particular WP or DTP package. It does not replace any vendor's manuals, but is a useful complement to them. Vendors of packages could consider including a copy of the book with their distribution materials in an attempt to boost the quality of resultant documents. Providers of advanced WP or DTP courses may find it useful as a reference book for the design aspects of their courses.

In summary, this book will be a useful reference for all users of WP or DTP packages who have gained at least a basic level of skill, are about to produce documents that will be used outside their immediate workgroup, and who want to maximise the readability and effectiveness of their work.

Steve Munro
Bureau of Meteorology, Melbourne


This book is one of a series of six (so far) published by the Open University Press on the topic of Robotics. Others in the series deal with a number of industrial applications, and also robot sensors. The current book deals only with the application of current Machine Intelligence techniques to the problems of industrial robots. It concentrates on the use of sensor information to decide what to do, rather than how to do it, with goal of providing more flexibility to robot systems.

The book is divided into eleven chapters, each with a summary and short bibliography. Each chapter acts as an introductory survey of its topic, with taxonomies given of the various techniques possible.

After an introductory chapter, there is a chapter on sensors generally, followed by two on machine vision. These are followed by three chapters covering aspect of Machine Intelligence: Knowledge Bases, Planning Systems and Expert Systems. There are also chapters on Speech Processing, Error and on System Design.

The tone of the whole book is broad, introductory and conversational. It is profusely illustrated with diagrams, tables and examples. There is also an index of approximately 300 entries.

For computer science majors, who typically have little interest in kinematics or control systems, a course of 10-20 lectures based on this book as a text would be highly popular. By combining the two bandwagon topics of Robotics and Machine Intelligence in the one course, it would attract good enrolments. Professor Lee: you have a winner.

Don Herbison-Evans
University of Sydney


This book, the latest in this long series, provides surveys of five diverse areas of computing. First, Millen provides a discussion of computer security and models for representing systems with multiple security levels. The significant contribution of the Bell-LaPadula model is described. He emphasises the problems of translating formal models into specifications.

Carroll's contribution is in the area of human-computer interaction. He traces this field's evolution from laboratory evaluation of computer systems and interactions, via attempts to use psychological theory to describe human information processing, to the current focus on usability as a major criterion in designing the human-computer interface.

Liu gives an extended discussion of protocol engineering. This area is concerned with the design, validation and verification of protocols for use in data communications between widely separated entities. Liu discusses formalised methods like Petri Nets. There are also descriptions of heuristics which can be applied to attempt to reduce the effects of the 'state explosion' problem when verifying systems.

Newborn's article is entitled 'Computer Chess: Ten Years of Significant Progress'. The increased power of equipment and techniques has allowed the top-rated chess programs to improve from the ability of a
In reviewing a book about FORTRAN, it is necessary to remember that one is reviewing the book and not the language — not an easy task with such a good book about such a bad language!

This book is all about the latest standard for this widely used language. Since many readers may have only experienced FORTRAN in its earliest form, FORTRAN 66, it is worth mentioning that modern FORTRAN, i.e. FORTRAN 90, includes many of the features of other modern programming languages, such as IF-THEN-ELSE statements, structured data, operator-overloading, pointers, recursive procedures and dynamic data structures. Of course, nobody can be made to use these features, and backwards compatibility means that many of the undesirable features of FORTRAN remain. Anyway, as I said earlier, this is a review of the book, not the language.

The book is well written, well organised, and reasonably easy to follow. It doesn't assume any particular previous knowledge, and is equally well suited to FORTRAN and non-FORTRAN literate programmers alike. It does assume a general knowledge of programming, and so is not an "introduction to programming" style of book. It describes the features of the language clearly, and where appropriate, it comments on how they differ from other languages, and also how they differ from previous versions of FORTRAN.

The book has a comforting feeling of authority and common sense. Both authors were members of the ANSI committee X3J3 responsible for the standard, and so the accuracy of the material can be assumed to be excellent. The book includes useful insights into the history of the development of the FORTRAN 90 standard. The authors also make some useful categorisation of the language structures, including a section called "deprecated features" meaning features (new and old) which are best avoided in order to produce more reliable or understandable code. The only improvement to the book would be to include a copy of the standard itself as an appendix, which I would personally recommend for any future edition.

The book is well suited to those who need a reference book on FORTRAN 90, those who need to program in FORTRAN 90, or those who merely wish to gain an understanding of the language. It does include exercises after each chapter, suggesting that it is also aimed at the textbook market. I would certainly recommend it as a textbook for any course on FORTRAN as a second language. The pace and detail of the book mean that it would be difficult (but not impossible) to use it in a first level programming course. The price is sufficiently reasonable for it to be used as a set text.

Overall, I recommend the book highly.

Neil Bergmann
CSIRO/Flinders Joint Research Centre in IT


As the title indicates, the book describes and discusses the concepts and functions of IBM's VM operating system.

This book clearly states the intended readers are System Programmers that have limited experience working on VM, and those who are beginning to work on a VM system. The book is written so that someone with no previous VM experience can benefit from reading it.

The book is divided into six sections and each section is then divided into topics. The early sections introduce VM and the evolution of the VM operating system. For those prospective readers that have previous VM experience the first two sections of this book may appear to be very tedious and could be browsed, as they discuss introductory and background topics of the VM operating system. The following sections begin to discuss the functions of VM and tuning considerations.

One of the main features of this book is the way it discusses the tuning options that are a feature of VM. When a tuning feature is presented both the benefits and the trade offs are both discussed. As indicated in the title, this book is only intended as a beginners tuning guide and as such many of the tuning options are not discussed to the depth that may be required when tuning or installing a VM system. At times when discussing tuning options the use of graphic examples (diagrams) could have provided the reader with a better understanding of the concept being presented.

When reading this book it was quite apparent that on occasions there were some anomalies in what was being stated. The majority of these appeared to be oversights in the editing of the book. An example of this can be found on page 88 when discussing TVRATIO.

The TVRATIO is used as a rough and ready measurement of the VM overhead. It is the ratio of TTIME Against VTIME. VTIME is the total length of time taken for application to process, i.e. time spent in problem state and supervisor state. VTIME is the time that the application code itself was processing, i.e. the time spent in problem state. The difference between TTIME and VTIME is the length of time spent in supervisor state.

In this excerpt it appears that VTIME and TTIME have been misrepresented which may confuse the reader.

Introduction to VM: A Beginner's Tuning Guide could provide System Programmers with limited VM experience a basic grounding in the tuning options and concepts relating to VM.

(This review represents my personal views and not those of the ANZ Banking Group.)

Stewart Pitt
ANZ Banking Group


It is claimed by the authors that this book represents the first to focus exclusively on the discrete cosine transform (DCT) and its applications. The book is aimed at the graduate student level but would make a good reference book on the subject for engineers involved in digital signal processing (DSP) applications. Indeed, the book has a practical approach, oriented towards implementations (particularly through VLSI and Application Specific ICs) and applications. Although not expensive, the book would be a worthwhile addition to the professional libraries of students,
researchers and engineers interested in fast DSP algorithms.

The book consists of seven chapters and an extensive list of appendices. The first two chapters introduce the DCT and discuss its motivation via the discrete Fourier transform and describes some of its properties. Chapter 3 considers the relationship to the Karhunen-Loeve transform (KLT) and discuss the DCT as an approximation to the KLT for a class of first order Markov processes.

Chapter 4 considers a number of approaches to the generation of fast algorithms for the DCT, including decimation in time, decimation in frequency and matrix factorisation methods. Chapter 5 considers the extension of two dimensional DCT algorithms. The performance of the DCT is the subject of the following chapter. Performance indicators considered here include rate distortion efficiency (for coding), residual correlation (for whitening), and least squares residuals (with reference to Wiener filtering). Chapter 7, being over 200 pages in length, is the larges of the book. It focuses on a wide range of applications of the DCT, including filtering, transmultiplexers (for FDM — TDM conversion), speech coding and analysis, data conversion and transform coding. Particular attention is given to image coding and video applications which are topical due to the fact that the DCT has been accepted as the standard method for image coding by the CCITT. The appendices contain FORTRAN and C programs for some of the DCT algorithms considered in the test proper. There are also listings of DCT chip and hardware manufacturers and brief product descriptions, something I consider to be an excellent addition. The book concludes with an extensive categorised bibliography which is current to mid-1990.

The book is written in a straightforward style with considerable attention to various approaches and related topics which have appeared in the literature. Many figures and block diagrams are included and help considerably in the understanding of the various applications of the DCT. I also found the application chapter to be extensive and a useful guide to some of the system approaches for image coding and conversion. Some colour and grey plates are also included.
small-systems programmers and designers as a starting point in identifying
the primary functions of the common commercial systems outlined. It
would also be useful as a reference text for a first course in computing for
business students. Business professionals who are end users of informa-
tion systems would probably gain the greatest benefit from this book.

Exploration in Information Systems Development. Blackwell Scienti-
cific Publications, Melbourne, 280pp., $49.95 (paperback).

When I first saw the title of this book, I thought it was a text on some new
piece of software in the Hypertext vein. However once having received the
book, I found out that it was a about a new method of systems analysis
that takes a holistic approach to software development rather than just
concentrating on particular parts of the analysis and design methodology.

The multiview methodology looks upon systems analysis as encompass-
ing five interrelated stages: analysing human activity in the existing
or proposed system, analysing the information flows, a study of the
socio-technical aspects, the human computer interface and the design of
the technical aspects. In doing this it attempts to form a system that is
complete in both human and technical terms.

The book consists of 19 chapters divided seven parts. Part 1 gives an
introduction to Multiview and consists a brief chapter on Information
Systems Development, establishes a framework definition for informa-
tion systems and introduces the first case study. Part 2 consists of three
chapters looking at the human activity: one chapter is devoted to develop-
ing a rich picture of the system. A rich picture is a pictorial representa-
tion of the organisation. This is used as a fundamental building block for
the remainder of the methodology. The next chapter looks at developing a
root definition (a statement of the problem and the human activity for the
system). Chapter six goes on to explain how to build and test the concep-
tual model for the system based on the root definition developed in the
previous chapter. In the final chapter of this section the second case study
is introduced.

Having developed the basic definition of the activity, Part 3 then
spends two chapters looking at a detailed analysis of the information
flows by examining events and data flows within the system (chapter eight)
and analysing the entities (chapter nine). Chapter ten introduces the
third case study.

Part 4 then goes on to consider what the authors refer to as the
socio-technical aspects of the system. I believe that this is one of the better
aspects of the Multiview method in that it asks the analysts to examine in
conjunction with the users how the proposed system can be made com-
patible with their needs and how it can be developed so as to fit comforta-
ably into their working lives. The authors are at pains to point out that
consulting the user does not just involve “doing a public relations job to
placate the staff”!, instead it involves thinking about and discussing the
realistic alternatives and allowing the problem owner to make informed
decisions about which alternative to adopt.

Part 5 consists of three chapters, the first of which takes an all too brief
look at the human computer interface. I find this very surprising in a book
that purports to be interested in developing systems that the user will be
happy using — ten pages including some poor examples of menu systems
is all that is devoted to this aspect. Chapter fourteen examines some
strategies for design or how an implementation may be developed that
meets the identified requirements. It includes some discussion of how and
when to select a computer package, when to use prototyping but leaves
much of the discussion of this until chapter sixteen. Another case study
rounds of this part of the book.

In Part 6 we at last come to a chapter that covers the detailed design of
technical aspects of the system to carry out all the tasks identified
previously. The output from this section will be an Information System
Specification and in chapter seventeen aspects of acceptance testing,
maintenance and development are considered. Finally this part contains

the last case study which the authors have described in a separate
publication, so the reader only gets a précis of the problem (I found it
rather lacking in detail) and a description of what arises after the Multi-
view method is applied to it. This case study has also formed the basis of a
final year project for students undertaking a Computer Science degree
and much of the detail contained in this chapter has been summarised
from their work.

The final part consists of a brief chapter summarising lessons learnt
from using the Multiview methodology in the field. It is a worthwhile
addition in that it gives some pointers to how flexible this method can be
and some of the pitfalls that one can strike in this holistic approach.

I am not altogether happy with this book as a means of illustrating the
usefulness of and method by which Multiview can be used to construct a
good working system that everyone can be happy with. I feel that the
methodology has much to recommend it, but the authors should have put
more effort into clarifying and developing illustrations of its use. There
are a number of case studies throughout the book — this number could
have easily been halved and more effort put into explaining and develop-
ning these. The book also has an annoying habit of introducing a topic and
then saying that discussion of this will be deferred until later in the book.
Surely if it was important enough to raise at that point then it is important
eough to discuss then. One aspect that did annoy me was in the exercises
at the end of chapter nine it refers the reader to a case study that is to be
introduced in the next chapter. Surely for a text that is about logical
design it would have been better to introduce the case study and have the
questions about it in the same or a subsequent chapter.

This book would provide interesting reading for students who have
already spent some time studying systems analysis or for practising
analysts who would like a text introducing one of the newer develop-
ments in their field; however be warned, it does eschew the use of any of
the standard symbols developed for analysis and design.

I cannot recommend that you buy this book; maybe persuade your
institute library to purchase it instead.

Andrew Wenn

WEISKAMP, K. and FLEMING, B. (1990): The Complete C++ Primer,

Weiskamp and Fleming have put together an excellent introduction to
C++. The text introduces C++, then OOP (Object Oriented Program-
ing). Next, it covers classes, including data hiding, friend functions, and
the creation and initialisation of objects. Contracts with C are made.
Virtual functions are described. C++ functions are described next, and a
full chapter is devoted to working with them. Polymorphism is
introduced. Operator overloading is then described in detail, and con-
structor and destructor functions are explained and illustrated. Inherit-
ance and class hierarchies, including multiple inheritance, are followed
by stream I/O and record-oriented I/O. The final chapter develops a
complete simulation for an automobile control centre, i.e., speedometer,
odometer, and accelerator controls. Since the main text was developed
around version 1.2, an appendix includes the new features and changes
that surface in version 2.0.

On my first glimpse of the text, I was envious of the writers. They have
created a lively and lucid text that is an easy read. Students will enjoy the
pace and spirit of the material.

What makes it a good text? The student is programming within the
first few chapters. Each chapter has a wealth of complete and executable
examples. Contrasts with C are included as appropriate. Notes of interest
and points of emphasis are offset in the text. There’s no focus on a
particular vendor’s implementation of C++. The closing simulation
brings it all together. The record I/O makes it attractive to business
as well as to CS.

Its shortcoming? Only one, and that not a major concern. Not enough
questions an programming exercises at the end of each chapter for the
student to explore!

Robert M. Lynch
University of Northern Colorado

40 THE AUSTRALIAN COMPUTER JOURNAL, VOL. 24, No. 1, FEBRUARY 1992
NEW LIBRARY SYSTEM FOR VICTORIA UNIVERSITY OF TECHNOLOGY
Victoria University of Technology has chosen INNOPAC as its second-generation library system, replacing both a URICA system and a Dynix system.

The $500,000 system was installed in January and is expected to be fully operational at the beginning of semester in March.

The 100 terminals will be located at the University's four campuses.

The University Librarian, Doreen Parker, cited INNOPAC's functionality as the most important factor in the library's decision. 'INNOPAC was rated exceptionally good in acquisitions, serial control, materials booking, OPAC, and management information,' Ms Parker said.

'The quality of management information was particularly important given that we are a new University library formed from the Footscray Institute of Technology and Western Institute libraries. There will be a great need for accurate information about the collections and their use to assist in planning the reorganisation and expansion of the libraries to meet the University's teaching and research needs,' Ms Parker said.

'The INNOPAC materials booking module was also of particular interest,' Ms Parker said. 'Our libraries are responsible for supply of audio-visual equipment to lecture theatres and seminar rooms throughout all buildings on the campuses.'

Victoria University's 100 user INNOPAC will be mounted on a MIPS Computer Systems RC3330 computer, a RISC-based processor. It will run under the UNIX operating system, providing the best factor in the library's decision. 'INNOPAC is based on international communications and hardware standards, so OPAC access can be easily provided to staff and students in their offices and departmental computer laboratories via local area networks on each campus and inter-campus telecommunications links,' Ms Parker said.

SEMINAR ON THE DEVELOPMENT OF PRIVATE NETWORKS
Standards Australia, in conjunction with AUSTEL, is conducting important one-day seminars dealing with the development of private networks.

Private networks are a key element in improving the efficiency of corporate communications. They eliminate per-call charges, minimise network congestion, and offer an ever-expanding range of other communications options. The recent publication of the AUSTEL 'Private Network Design Guide' is an important step in facilitating the design and implementation of these networks.

The seminars are aimed at every person and organisation involved in the planning, design, operation and upgrading of private networks. They will present all the issues which must be considered in assessing the viability of a private network, and the major design criteria.

Beginning with an introduction to the reasons for private networks and their economies, the seminars will cover the design considerations addressed to the Design Guide and conclude with carrier considerations. They will build on the previous seminar dealing with the draft Code of Practice and will include case studies.

The expert speakers will also discuss the revision of the AUSTEL technical standard covering general requirements for interconnection of private networks with the public switched telephone networks (TS 012), and the principal proposed changes, which will have an important impact on the majority of private networks. These seminars will provide the ideal opportunity to consider the ramifications of the planned changes and discuss them with the participants.

Seminars are to be held in Melbourne (4 March), Canberra (6 March), and Sydney (11 March). For more information and registration details, contact the Seminar Secretary, phone (02) 963 4111, fax (02) 959 3896.

SECURITIES COMMISSION FUNDS IMAGE COMMUNICATIONS RESEARCH
The Australian Securities Commission (ASC) is about to invest $0.5 million into image communications research. For the past year, it has been operating a world-leading corporate document imaging system.

This 'state of the art' system is based at the Image Processing Centre (IPC) at Morwell, where corporate documents are scanned, optically stored/retrieved and then transmitted via Telecom ISDN to ASC offices throughout Australia.

With the aid of a consultancy with RMIT, headed by Professor Ian Bates and Mr Jeff Lewenberg, the IPC system was specified, designed and implemented within both budgetary and time constraints.

The ASC is now looking to future expansion and advancement of this technology. The new funding will be spread between researchers in Melbourne and the Latrobe Valley. The Melbourne group is headed by the RMIT Centre for Advanced Technology in Telecommunications (CATT) under the directorship of Associate Professor Lindsay Jackson.

CATT will work closely with the Intelligent Decision and Control Systems group, both located within the Collaborative Information Technology Research Institute (CITRI) which is a joint initiative of RMIT and the University of Melbourne.

The planned work covers a number of different aspects of image scanning, storage and transmission/networking. Advanced network management for ISDN networks and integrated Local Area Networks (LANs) with the eventual aim of providing dynamic reconfiguration and optimisation, is a topic for the CATT researchers. The current image transmission utilises six 'B' channels within the available 30 in primary rate ISDN (Telecom's Macrolink). Telecom is naturally interested in these developments and is also supporting the work.

Other CITRI research will lead to more efficient storage and access of the images stored on optical disc and often known as WORM technology (Write Once Read Many times). This equipment has been provided by Kodak who is also supporting the work.

The scanning of original documents is an important activity of the IPC and hence the ASC funding also covers research in this area being carried out by a team Monash — Gippsland. The team, under the leadership of Professor Ken Spriggs, is concentrating on advanced optical character recognition techniques.

The research outcomes should lead to a number of unique enhancements to the ASC system which already is seen throughout the world as a pioneering effort, particularly in the application of ISDN.

THE NEW GAS BILL IN FOCUS
The Gas and Fuel Corporation in Victoria has offered the community a progressive approach to conservation, by actually giving the consumer the facts on gas consumption on the bill in easy to understand graph format, which compares the gas account over a four-month period, allowing for a direct comparison on two billing periods over two years.

The new bill was developed following extensive research with the consumer and the Customer Services personnel at the Gas and Fuel, and using the services of Moore Business Systems, a company who has extensive skills in all aspects of forms management control, delivering to all concerned an easy to read, easy to process billing system.

All 1.2 million Gas and Fuel customers have now seen the new Gas account.

The computer programming of the form design was handled by Moore Business Systems. The system consisted of a number of module programs (16 in all) as well as two major programs, 40 Gas and Fuel specific subroutines and eight Moore general subroutines. This amounted to 10,800 lines of FORTRAN code, with approximately 133 variables and 3100 subroutines and eight Moore general subroutines.

Other CITRI research will lead to more efficient storage and access of the images stored on optical disc and often known as WORM technology (Write Once Read Many times). This equipment has been provided by Kodak who is also supporting the work.

The scanning of original documents is an important activity of the IPC and hence the ASC funding also covers research in this area being carried out by a team Monash — Gippsland. The team, under the leadership of Professor Ken Spriggs, is concentrating on advanced optical character recognition techniques.

The research outcomes should lead to a number of unique enhancements to the ASC system which already is seen throughout the world as a pioneering effort, particularly in the application of ISDN.
entry and customer service personnel), and external operations of the G and F (Australia Post and ICL equipment). The customer requirements were decided upon following market research.

The bill needed to be easy to understand — easy to read — size and style of typeface and also the total dollar amount needed to be clearly seen, as well as the due date for payment. People also wanted itemised billing as shown by the phone enquiry level in this area, clearly seen, as well as the due date for payment.

Clear instructions on payment methods were also wanted, and clear concise information on locations where bills could be paid.

The Computer Information for the ICL equipment at the showrooms as well as the code keyed in by Australia Post also needed to be isolated from other information but clear and concise.

The bill was designed to be laser printed in black for optimum clarity and the base stock printed in blue, highlighting the billing amount area as well as the ‘pay by’ date for the speedy payment to the Gas and Fuel.

Marketing messages can also be added, purpose selected depending on the customer profile and usage.

Billing amounts and concession eligibility can also be purpose written.

The graph information differentiates the current usage from the past 12 months, giving a clear indication of changes in usage — even a cost per day analysis is available. Each graph is purpose designed by the computer, giving the flexibility of scales for different usage quantities. Giving the consumer the clear indication of changing the usage — and conserving the resource where possible.

CSIRO — SPREADING THE WORD WITH PROGRESS

One of the most difficult tasks for publishers of non-fictional books, often referred to as textual or bibliographical publications, is the compilation of the index. This is just one area where the CSIRO have reported major productivity gains since using the Progress Application Development Environment.

One group of the Management Information Systems Branch of CSIRO is responsible for the production of scientific and technical bibliographies and directories on behalf of CSIRO and related government bodies. Typically 10 major publications (some in excess of 500 pages) are produced annually, along with a number of smaller compendiums and directories. Each has to be keyed, indexed, typeset and printed.

According to Barry Cheney, Manager of the Systems Design Unit in Melbourne, 'Until three years ago we used the CSIRONET network for the production of publications. It was a batch-based Fortran system and it typically took three to four weeks to get a document to the printing stage after data entry. Indexing was a nightmare which could take weeks to get right. Since we've introduced the on-line Progress system it now takes an hour from the time an editor has finished editing text until final camera-ready copy is ready to go to the printers! To compile subject indexes using the Fortran system we used three programs and several hundred lines of code and still had to do several sorts and merges. This was replaced with four lines of Progress code and for many of the publications it is a matter of pressing one key.'

Cheney continues, 'It was an easy transition for us from Fortran to Progress. It is surprisingly easy to use and quite revolutionised the way we do things.'

'In many ways we were amazed at the versatility of Progress. Our needs were not the norm — for example we do not have numerous small coded fields such as you may find in an accounting or manufacturing application, some of our fields may be thousands of characters long. Because of the scientific nature of the material we deal with we have needs for special characters, superscript and subscripts, and italics, and we have found ways of accommodating these within Progress.'

Some 30 publications are handled by the group. Each publication has a dedicated Progress database into which the text is entered. The publication editor then verifies and edits the subject matter, and identifies the appropriate information for indexing and cross-referencing. The whole publication, from contents page to indexes, is automatically generated (using whatever parameters requested) and the output can be seen on the screen, or sent, using TeX typesetting software, directly to either a laser printer for a proof copy or to a VariTyper 600 typesetter for final typesetting. The output from the laser printer is camera-ready copy which is sent to the printers for production of the publication.