SPECIAL ISSUE ON INFORMATION TECHNOLOGY AND THE LAW I

113 Guest Editorial — Intellectual Property in IT Artefacts

114 A study of unauthorised software copying among post-secondary students in Hong Kong
   G WONG, A KONG, S NGAI

123 The lessons of recent EC and US developments for protection of computer software under Australian Law
   P WATERS, PG LEONARD

133 Intellectual property protection of user interfaces in USA and Germany
   A WIEBE

140 Copyright and computer software — Autodesk Inc. v Dyason and Kelly
   JWK BURNSIDE

SPECIAL FEATURES

122 New reviewers required and update of records

145 Referee and reviewer profile

147 Book Reviews

Published for Australian Computer Society Incorporated
Registered by Australia Post, Publication No. NBG 1124
The Australian Computer Journal is an official publication of the Australian Computer Society Incorporated.

CONTRIBUTIONS
All material for publication should be sent to:
Mr Rob Cook, Editor
Australian Computer Journal
Centre for Information Technology Research
University of Queensland
St Lucia, QLD 4067

Prospective authors may wish to consult manuscript preparation guidelines published in the February 1986 issue. The paragraphs below briefly summarise the essential details.

Types of Material: Four regular categories of material are published: Papers, Short Communications, Letters to the Editor and Book Reviews.

Generally speaking, a paper will discuss significant new results of computing research and development, or provide a comprehensive summary of existing computing knowledge with the aim of broadening the outlook of Journal readers, or describe important computing experience or insight.

Short Communications are concise discussions of computing research or application.

A letter to the Editor will briefly comment on material previously appearing in the Journal, or discuss a computing topic of current interest.

The term 'computing' is interpreted broadly to include descriptions of computer hardware and software, and major computer applications such as information systems, expert systems, computer networks and office automation.

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

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

Format: Papers, Short Communications and Book Reviews should be typed in double spacing on A4 size paper, with 2.5 cm margins on all four sides. The original, plus two clear bond paper copies, should be submitted. References should be cited in standard journal form, and generally diagrams should be ink drawn on tracing paper or board with stencil or Letraset lettering.

Papers and Short Communications should have a brief Abstract, Key word list and CR categories on the leading page, with authors’ affiliation as a footnote. The authors of an accepted paper will be asked to supply a brief biographical note for publication with the paper.

Copyright © 1990 Australian Computer Society Inc.

Production Management:
Associated Business Publications Pty Ltd, Room 104, 3 Small Street, Ultimo NSW 2007 (PO Box 440, Broadway, NSW 2007). Telephone 212 2780, 212 3780.

All advertising enquiries should be referred to the above address.

Copyright © 1990 Australian Computer Society Inc.

Subscription/Change of Address Form
Photocopies of this form are acceptable.

NAME .................................................................

CURRENT ADDRESS ......................................................
..........................................................................................

☐ Please enrol me as a subscriber for 1990. I enclose a cheque or bank draft for $40.00.

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

*ATTACH LABEL HERE

Send all correspondence regarding subscriptions to the
Circulation Manager, PO Box 319, Darlinghurst, NSW, 2010, Australia.

Editor
R COOK University of Queensland

Associate Editors
MJ LAWRENCE, University of NSW
C ANDREWS, DN WILSON, University of Technology, Sydney

Editorial Committee
J LIONS, JM BENNETT, RA CLARKE, PC POOLE, AY MONTGOMERY, AJH SALE, CK YUEN

Office Bearers
BA UNDERWOOD President
P MURTON, GW DOBER Vice-Presidents
JA GODDARD Immediate Past President
RG HEINRICH National Treasurer
Executive Director & Chief Executive Officer
PO Box 319, Darlinghurst, NSW, 2010
Telephone (02) 211 5855

Publication Board
JM HUGHES Governor

SUBSCRIPTIONS
The annual subscription is $40.00. All subscriptions to the Journal are payable in advance and should be sent (in Australian currency) to the Australian Computer Society Inc.
PO Box 319, Darlinghurst, 2010
A subscription form may be found below.

Price to Non-Members: There are four issues per annum. The price of individual copies of back issues still available is $9.00. Some are already out of print. Issues for the current year are available at $12.00 per copy. All of these may be obtained from the above address. No trade discounts are given, and agents should recover their own handling charges.

Members: The current issue of the Journal is supplied to personal members and to Corresponding Institutions. A member joining partway through a calendar year is entitled to receive one copy of each issue of the Journal published earlier in that calendar year. Back numbers are supplied to members while supplies last, for a charge of $9.00 per copy. To ensure receipt of all issues, members should advise the Branch Honorary Secretary concerned, or the National Secretariat, promptly, of any change of address.

Membership: Membership of the Society is via a Branch. Branches are autonomous in local matters, and may charge different membership subscriptions. Information may be obtained from the following Branch Honorary Secretaries. Canberra: GPO Box 446, Canberra, ACT, 2601. NSW: 6th Floor, 66 King Street, Sydney, NSW, 2000. Qld: Box 135, GPO, Aspley, Qld, 4034. SA: Box 2423 GPO, Adelaide, SA, 5001. WA: Box F320 GPO, Perth, WA, 6001. Vic: PO Box 319, Darlinghurst, NSW, 2010.

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

Intellectual Property in IT Artefacts

That complex of computing, telecommunications and robotics, commonly referred to as Information Technology, is having an increasingly significant impact on the world. As a result, the benefits and the costs of IT are becoming increasingly worth fighting over. This Special Issue was conceived in order to stimulate consideration of the issues, both by authors, and by readers.

And stimulate it did. The Call for Papers attracted 30 offers, resulting in 15 papers from 6 countries. The Editors considered that ten of these were, subject to modification, appropriate for publication in the Journal. Unfortunately the space available forced that list to be reduced to just four papers in November 1990 and three in February 1991. It was decided that, rather than selecting papers which reflected the enormous range of issues in the field, it would be appropriate to focus the Issue on two dominant themes; which were handled by two different editorial teams.

The Editors expressly sought to bridge both the gap between the information technology and legal disciplines, and the all-too-familiar gulf between researchers and practitioners. We believe we have been successful in that regard. One of the papers is by a team of computer science and information systems academics; two are by academics who are dual-qualified in computer science and law; two are by teams comprising IS or computing academics and a practising lawyer; and two are by legal professionals who practice in the computers and law area.

The Editors commend the papers to the Journal's readers, at all points on the practitioner/academic spectrum.

It is generally accepted that progress in IT, as in any technology, is contingent upon innovators being reasonably assured of receiving a return on their investment. Hardware is protectable by way of patent, design and (since the mid-1980s) circuit layout law, but for software developers the primary means of gaining a small monopoly space is through copyright. Copyright is a long-standing branch of law, which is very rich in case law and erudite arguments, and the scope for courts to interpret the applicability of copyright law to software is enormous.

This Issue contains four papers dealing with different aspects of software copyright. The first, Wong, Kong & Ngai 'A Study of Unauthorized Software Copying Among Post-Secondary Students in Hong Kong', investigates the nature and extent of that phenomenon. It establishes that many students not only use copied software, but also justify their actions to their own satisfaction. It is likely that the results are at least to some extent generalisable, fairly probably to the Hong Kong workforce, and possibly to both students and the workforce in many other countries.

The other three papers address current issues in software copyright law. Waters & Leonard's 'The Lessons of Recent E.C. and U.S. Developments for Protection of Computer Software Under Australian Law' considers the scope of copyright protection, and analyses the current position in law and policy in Europe and the United States. The questions of reverse engineering and user adaptation of software are also discussed, and implications for Australian importers, developers and users assessed. Although weaknesses and inconsistencies in international developments are identified, Australia's ability to influence the directions of software copyright law are argued to be limited.

Wiebe's 'Intellectual Property Protection of User Interfaces in the U.S.A. and Germany' examines one specific aspect of software copyright. There has been a tendency in the United States for the 'look and feel' of software to be regarded by the courts as being worthy of copyright protection, and this is influencing the courts' interpretations of copyright law in other countries. Although the paper concentrates on the U.S. and German contexts, the IT industry is so international in flavour, and copyright law sufficiently similar throughout the world, that the analysis has a great deal of relevance to ACJ readers in all countries.

Burnside's 'Copyright and Computer Software: Autodesk Inc. v. Dyason & Kelly' reviews the first case which has been decided since the 1984 amendments to the Commonwealth Copyright Act. Those amendments explicitly extended the scope of copyright law to include computer programs, and were made after the Apple v. Computer Edge case had created serious doubt as to whether copyright law had ever applied to software in Australia. The judgment at first instance in the Autodesk case appeared to have greatly modified conventional notions of software copyright, but has been (very recently) reversed on appeal. Nonetheless, the decisions, and the arguments supporting them, are of great importance to the IT industry. Burnside argues for a new form of protection, specifically tailored for software.

These papers provide valuable and up-to-date information on the vexed question of software copyright. That the authors' interpretations and conclusions are not entirely consistent with one another serves to underline the difficulties which face the courts and the legislature, and hence IT suppliers and users.

Roger Clarke
Reader in Information Systems
Department of Commerce
Australian National University

Graham Greenleaf
Senior Lecturer
Faculty of Law
University of N.S.W.

The Editors

THE AUSTRALIAN COMPUTER JOURNAL, VOL. 22, NO. 4, NOVEMBER 1990 113
This study investigates software copying among post-secondary students in Hong Kong. Over 500 students were surveyed in 1989. The magnitude of the problem in Hong Kong is quantified. Using factor analysis and regression analysis, several factors that significantly affect three variables (use of copied software; allowing others to copy software; and purchase of illegal software) are identified. Implications of the findings, and recommendations on how to control the problem are presented.

Keywords & phrases: software copying, software piracy

CR Categories: K.4.2, K.5.1

1. INTRODUCTION
Unauthorized software copying in Hong Kong has reached epidemic proportions. Between August 1, 1989 and March 31, 1990, over U.S. $6.5 million worth of illegal software, manuals, and equipment was seized by customs officials (South China Morning Post, Apr. 2). In schools, students openly exchange software that has been copied without authorization. Some of them even ask their instructors to make arrangements for them to duplicate software and manuals. In business, local software distributors estimate that revenues would multiply ten-fold or more if corporate copying of software were eliminated (Au, 1989). These all happen despite copying software without authorization being prohibited by law (U.K. Copyright (Computer Software) Amendment Act 1985; Extension to Territory Order 1987).

Some users have argued that software copying is good for society. They claim that by copying software, users who otherwise could not afford to buy the program can gain access to the technology. As a result, the society will become more competitive. On the other hand, software copying causes many problems for many people. Firstly, software copying is the most important channel for spreading computer viruses. Copied programs may be less expensive, but the potential losses caused by computer viruses such as loss of data; loss of business; loss of productivity; etc., may far outweigh the benefit. Secondly, software copying leads to higher prices for legitimate users because publishers have to recover R&D expenses. Also, software publishers may be less willing to do business in an area where software copying is serious. This will slow down the spread of advanced technologies and make the region less competitive. Alternatively, publishers may implement copy-protection mechanisms to slow down copying. This will in many cases cause legitimate users inconvenience and loss in productivity. Finally, the costs associated with law enforcement and legal actions are high. The Hong Kong Customs & Excise Department has a 44-man task force just to work on stopping the sales of illegally copied software. In addition, hundreds of police are involved in every raid. Subsequently, the Legal Department (responsible for prosecution) and the Courts (responsible for the trial) are involved in legal actions. The taxpayers of Hong Kong have to pay for these costs.

Clearly the problem must be addressed. Yet, many important questions related to software copying are unanswered. It is known that many users copy software, but the actual magnitude of the problem is not known. We do not know what are users' attitudes towards software copying; and what makes computer
users copy software or purchase illegally copied software. In order to develop more effective strategies to combat the problem, these issues must be addressed.

This study investigates software copying among post-secondary students in Hong Kong. Specifically, it addresses three important questions:

1. What is the magnitude of the problem among post-secondary students?
2. What attitudes do students have towards unauthorized software copying?
3. What are the factors that affect software copying among post-secondary students?

Results from this study will give policy makers and educators more insight into the problem, so that better strategies in dealing with the problem can be developed.

Post-secondary students are selected for this study because they are a major group of computer users in Hong Kong. They are also known for copying software. Furthermore, it may be much easier trying to educate students than trying to influence working adults. After all, school is where students learn how to be professionals. If we cannot stop students from copying software in school, it will be much more difficult stopping them after they graduate.

2 PRIOR STUDIES

A number of studies on computer crimes have been conducted over the years. Krauss and MacGahan (1979) found that most computer crimes had been committed by people who had not previously been involved in criminal activities. In a report published by the American Bar Association (ABA 1984), seventy-two firms had reported losses of above U.S. $500 million, and 25% of major U.S. firms were uncovering serious incidents of abuse each year. Straub (1987) studied computer abuse in 1,211 DPMO organizations and found that in general, security countermeasures were effective in reducing computer abuse. However, all these studies had concentrated on computer crimes in the mainframe, large corporation environment, and had failed to deal with the problem of software copying which mainly occurs at the PC level.

Recently, two studies of software copying have been conducted in North America. Shim and Taylor (1988, cited in Lin 1989) found that 70% of business faculty members had admitted copying software, and Lin (1989) found that 56% of all PC users surveyed had made copies of someone else’s software, and 39% had let someone else copy their software.

These studies generated interesting results, but it is unclear whether their findings are applicable in other countries, such as Hong Kong, which is culturally very different from North America. Studies that focus directly on Hong Kong are needed to answer the questions posed in section 1.

3 RESEARCH METHOD

A survey was conducted in November 1989 at the City Polytechnic of Hong Kong. A questionnaire was administered to about 600 students taking classes offered by the Department of Computer Studies. The students were selected according to the following criteria. Firstly, CS0101 (Introduction to Data Processing) students were excluded because a majority of the students in that class lack prior computing experience and might not have been able to answer the questionnaire objectively. Then, roughly one-third of the remaining students were randomly selected for the study. The questionnaire was pilot tested on a group of students before being used.

The questionnaire had three parts. Part I contained questions that measured the respondent’s demographic characteristics. Part II measured how often the respondent used copied software, allowed others to copy their software; and purchased illegally copied software. Part III contained thirty-five statements that measured the respondent’s attitude towards copying software. The questionnaire is given in Appendix A.

4 RESULTS

Five hundred and four (504) completed responses were collected. SPSS+/PC was used for data analysis. The results are presented in this section. Implications of the results will be discussed in the next section.

DESCRIPTIVE STATISTICS

Table 1 summarizes the seven descriptive statistics collected: Major of Students, Age, Sex, Employment Status, Family Background, and Level of Computer Knowledge.

Tables 2 to 4 summarize statistics on the three aspects of software copying: using copied software, allowing others to copy software, and purchasing illegal software. The results are tabulated by majors. Table 5 is the correlation between these variables.

The correlation between Using Copied Software and Allowing Others To Copy One’s Software was very high. However, while significant at the .01 level, the correlations between Purchasing Illegal Software and the other two variables were much smaller. This suggests that using copied software and allowing others to copy one’s software go hand in hand, while the purchasing of illegal software is perceived quite differently.
Table 1: Demographic Statistics of Respondents

a) Majors:
- Business: 179 (35.5%)
- Science & Technology: 72 (14.3%)
- Computer Studies: 199 (39.5%)
- Social Science: 46 (9.1%)
- Missing: 8 (1.6%)

b) Sex:
- Male: 269 (57.4%)
- Female: 234 (46.4%)
- Missing: 1 (0.2%)

c) Age:
- Mean: 22.19
- Std. Dev.: 3.33

d) Own PC:
- Yes: 325 (64.5%)
- No: 179 (35.5%)

e) Employment:
- Full Time: 134 (26.6%)
- Part Time: 89 (17.7%)
- Not Employed: 281 (55.7%)

f) Computer Knowledge (Self-reported):
- Know Nothing: 20 (4.0%)
- Very Little: 226 (44.8%)
- Average: 204 (40.5%)
- Good: 51 (10.1%)
- Excellent: 3 (0.6%)

g) Family Background (Self-reported):
- Grassroots: 268 (53.2%)
- Middle Class: 215 (42.7%)
- Upper Middle Class: 18 (3.6%)

Table 2: How Often Does a Student Use Copied Software

<table>
<thead>
<tr>
<th>Science &amp; Technology</th>
<th>Social Studies</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td>(17%)</td>
<td>(15%)</td>
</tr>
<tr>
<td>Rarely</td>
<td>(2)</td>
<td>(10%)</td>
</tr>
<tr>
<td>(27% (25%)</td>
<td>(10%)</td>
<td>(15%)</td>
</tr>
<tr>
<td>Sometimes</td>
<td>(3)</td>
<td>(28%)</td>
</tr>
<tr>
<td>(51 (75%</td>
<td>(28%)</td>
<td>(15%)</td>
</tr>
<tr>
<td>Frequently</td>
<td>(4)</td>
<td>(20%)</td>
</tr>
<tr>
<td>(35 (20%)</td>
<td>(15%)</td>
<td>(10%)</td>
</tr>
<tr>
<td>All the Times (5)</td>
<td>(35 (20%)</td>
<td>7</td>
</tr>
<tr>
<td>(22 (13%)</td>
<td>(10%)</td>
<td>(15%)</td>
</tr>
<tr>
<td>Mean</td>
<td>3.09</td>
<td>4.28</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>1.26</td>
<td>.87</td>
</tr>
<tr>
<td></td>
<td>2.20</td>
<td>1.14</td>
</tr>
</tbody>
</table>

Table 3: How Often Does a Student Allow Others to Copy His/Her Software

<table>
<thead>
<tr>
<th>Science &amp; Technology</th>
<th>Social Studies</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td>(25%)</td>
<td>(7%)</td>
</tr>
<tr>
<td>Rarely</td>
<td>(2)</td>
<td>(5%)</td>
</tr>
<tr>
<td>(21% (29%)</td>
<td>(10%)</td>
<td>(5%)</td>
</tr>
<tr>
<td>Sometimes</td>
<td>(3)</td>
<td>(25%)</td>
</tr>
<tr>
<td>(52 (29%)</td>
<td>(10%)</td>
<td>(5%)</td>
</tr>
<tr>
<td>Frequently</td>
<td>(4)</td>
<td>(21%)</td>
</tr>
<tr>
<td>(37 (21%)</td>
<td>(20%)</td>
<td>(10%)</td>
</tr>
<tr>
<td>All the Times (5)</td>
<td>(37 (21%)</td>
<td>9</td>
</tr>
<tr>
<td>(22 (13%)</td>
<td>(10%)</td>
<td>(5%)</td>
</tr>
<tr>
<td>Mean</td>
<td>2.88</td>
<td>4.12</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>1.21</td>
<td>.89</td>
</tr>
<tr>
<td></td>
<td>1.14</td>
<td>1.14</td>
</tr>
</tbody>
</table>

Table 4: How Often Does a Student Purchase Illegally Copied Software

<table>
<thead>
<tr>
<th>Science &amp; Technology</th>
<th>Social Studies</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td>(60 (34%</td>
<td>(28%</td>
</tr>
<tr>
<td>Rarely</td>
<td>(2)</td>
<td>(19%</td>
</tr>
<tr>
<td>(62 (35%)</td>
<td>(19%)</td>
<td>(28%)</td>
</tr>
<tr>
<td>Sometimes</td>
<td>(3)</td>
<td>(26%)</td>
</tr>
<tr>
<td>(49 (28%</td>
<td>(26%)</td>
<td>(28%)</td>
</tr>
<tr>
<td>Frequently</td>
<td>(4)</td>
<td>(4%</td>
</tr>
<tr>
<td>(4 (2%)</td>
<td>(1%)</td>
<td>(5%)</td>
</tr>
<tr>
<td>All the Times (5)</td>
<td>(4 (2%)</td>
<td>(5%)</td>
</tr>
<tr>
<td>(3 (3%)</td>
<td>(3%)</td>
<td>(2%)</td>
</tr>
<tr>
<td>Mean</td>
<td>2.05</td>
<td>2.52</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>.95</td>
<td>.97</td>
</tr>
<tr>
<td></td>
<td>1.07</td>
<td>1.09</td>
</tr>
</tbody>
</table>

Table 5: Correlation Between Using, Copying, and Purchasing Illegal Software

<table>
<thead>
<tr>
<th></th>
<th>Use Copied Software</th>
<th>Allow Others to Copy Software</th>
<th>Purchase Illegal Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>1.0000</td>
<td>.7152</td>
<td>.3195</td>
</tr>
<tr>
<td>Allow Others to Copy Software</td>
<td>.7152</td>
<td>1.0000</td>
<td>.3252</td>
</tr>
<tr>
<td>Purchase Illegal Software</td>
<td>.3195</td>
<td>.3252</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

p < .01 for all correlations

ATTITUDES TOWARDS SOFTWARE COPYING

Part III of the questionnaire contained thirty-five statements measuring students' attitudes towards software copying. These statements were developed in several brainstorming sessions, where several instructors were asked to suggest reasons why students copy software. All statements were measured on a five point Likert-type scale (strongly disagree, disagree, neutral, agree, strongly agree). Because of the large number of statements involved, a factor analysis was performed to identify the underlying constructs.

Eleven factors were identified using Principal-Components Analysis and Quartimax rotation. Before rotation, these factors account for 53% of the total variation. However, only the first six factors, which have a cumulative percentage of 37.3, had meaningful interpretations. Factors seven through eleven were excluded from further analysis. The next section summarizes the six meaningful factors identified.

For each factor, the title represents the underlying construct interpreted by the authors. The Eigen Value is the total variation of that factor before rotation. Percentage of Variation is the percentage of variation in all 35 statements accounted for by this factor before rotation, and Cumulative Percentage is the percentage of variation explained by factors 1 through the current one before rotation. The statements loaded on the factor and the percentage of respondents who agreed or strongly agreed with the statements are also listed.
Factor 1: Attitude Towards Software Publishers

Eigen Value: 4.63 % of Var. 13.2

Statements Loaded On the Factor

- a) Software publishers don't want unauthorized copying because they are greedy. Agree or Strongly Agree
  - 26
- b) Software users should not have to worry about the interest of software publishers. Agree or Strongly Agree
  - 18
- c) Copying a software will actually make it more popular, and hence benefits the publisher. Agree or Strongly Agree
  - 53

Factors 2: Property Right

Eigen Value: 2.41 % of Var. 6.9

Statements Loaded On the Factor

- a) It is unethical to use software without paying for it. Agree or Strongly Agree
  - 29
- b) Copying software will hurt everybody in the long run. Agree or Strongly Agree
  - 23
- c) One should pay for goods and services received. Agree or Strongly Agree
  - 69
- d) Somebody spent a great effort to produce a software. It is unfair to copy it. Agree or Strongly Agree
  - 54

Factors 3: Acceptance of the behaviour

Eigen Value: 1.73 % of Var. 4.9

Statements Loaded On the Factor

- a) Copying software is O.K. because there is no penalty associated with doing so. Agree or Strongly Agree
  - 30
- b) I use unauthorized copies of software because my friends are doing the same. Agree or Strongly Agree
  - 38

Factors 4: Availability of Original Software

Eigen Value: 1.63 % of Var. 4.7

Statements Loaded On the Factor

- a) Original software is too expensive. Agree or Strongly Agree
  - 90
- b) It is more convenient to make unauthorized copies than to purchase the original one. Agree or Strongly Agree
  - 78

Factors 5: Commercial Use

Eigen Value: 1.41 % of Var. 4.0

Statements Loaded On the Factor

- a) It is O.K. to copy software as long as I do not use it for commercial purpose. Agree or Strongly Agree
  - 69

Factors 6: Concern for difficulty and cost associated with copying software

Eigen Value: 1.24 % of Var. 3.5

Statements Loaded On the Factor

- a) I will not copy software if there is penalty associated with doing so. Agree or Strongly Agree
  - 38
- b) I will not attempt to copy a software if it is heavily copy protected. Agree or Strongly Agree
  - 47

An interesting point to notice is that statements loaded on Factor 3 and statements loaded on Factor 6 are quite similar. There is no clear explanation for this. One possibility is that Factor 3 concerns mainly students' general feeling about software copying, while Factor 6 concerns students' intentions to do something. For example, a student may feel that copying software is not wrong (general feeling), but may also believe that it is not worthwhile to copy software if the penalty for doing so increases (intention).

Another interesting point is that the statement “Original Software is Too Expensive” does not belong to Factor 1. This suggests that the negative attitude towards publishers is not due to the high price of software.

REGRESSION ANALYSIS OF STUDENTS' BEHAVIOUR

Different dimensions of students' attitudes towards software copying have been identified by factor analysis. However, how students felt about the issue might or might not lead to the actual copying of software or buying of illegally copied software. To see how the two areas were related, a regression analysis was performed.

The independent variables included the 6 factors identified and the demographic variables listed in table 1. Dummy variables were set up for Sex and Major (Afifi and Clark, 1984). For Major, Business was selected as the basis, for Sex, Male was used as the basis.

The dependent variables were: How Often Do Students Use Copied Software, How Often Do Students Allow Others to Copy Software, and How Often Do Students Purchase Illegally Copied Software. Stepwise regression was used to identify variables that affected students' behaviour (Afifi and Clark, 1984). Stepwise regression selects a subset of independent variables that will yield an optimal prediction with as few terms as possible. Tables 7 to 9 summarize the results.

All three regression models had a significant R Square. However, the R Square for Purchasing Illegally Copied Software was much smaller than the other two (13% vs. about 38%). This suggests that Purchasing Illegally Copied Software was less affected by the independent variables studied.

The Use of Copied Software: Age, Sex, Employment Status, and Family Background all had no effect on the Use of Copied Software at the .05 level. These variables were not selected in the Stepwise Regression. However, those who owned a PC and those who had more knowledge of computers would more likely use copied software. Also, when compared with Business majors, Science and Technologies majors and Computer Studies majors would more...
### Table 7: How Often Does a Student Use Illegally Copied Software

<table>
<thead>
<tr>
<th>Variables Selected (PIN = .05)</th>
<th>B</th>
<th>SE</th>
<th>Beta</th>
<th>T</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Own PC</td>
<td>.58648</td>
<td>.11508</td>
<td>.21590</td>
<td>5.096</td>
<td>.0000</td>
</tr>
<tr>
<td>Computer Knowledge</td>
<td>.18467</td>
<td>.07297</td>
<td>.08465</td>
<td>2.531</td>
<td>.0117</td>
</tr>
<tr>
<td>Science &amp; Technologies Major</td>
<td>.39734</td>
<td>.14585</td>
<td>.10690</td>
<td>2.674</td>
<td>.0078</td>
</tr>
<tr>
<td>Computer Studies Major</td>
<td>.69070</td>
<td>.12298</td>
<td>.20520</td>
<td>5.617</td>
<td>.0000</td>
</tr>
<tr>
<td>Acceptance of the Behaviour</td>
<td>.17646</td>
<td>.04641</td>
<td>.01486</td>
<td>3.846</td>
<td>.0000</td>
</tr>
<tr>
<td>Access to Legal Software</td>
<td>.37708</td>
<td>.04949</td>
<td>.01401</td>
<td>7.551</td>
<td>.0000</td>
</tr>
<tr>
<td>OK if Not Making Money</td>
<td>.10078</td>
<td>.04786</td>
<td>.01401</td>
<td>2.106</td>
<td>.0358</td>
</tr>
</tbody>
</table>

### Table 8: How Often Does a Student Allow Others to Copy His/Her Software

<table>
<thead>
<tr>
<th>Variables Selected (PIN = .05)</th>
<th>B</th>
<th>SE</th>
<th>Beta</th>
<th>T</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Own PC</td>
<td>.66129</td>
<td>.11509</td>
<td>.24072</td>
<td>5.746</td>
<td>.0000</td>
</tr>
<tr>
<td>Computer Knowledge</td>
<td>.14923</td>
<td>.07447</td>
<td>.08645</td>
<td>2.004</td>
<td>.0457</td>
</tr>
<tr>
<td>Science &amp; Technologies Major</td>
<td>.66521</td>
<td>.14868</td>
<td>.17697</td>
<td>4.676</td>
<td>.0000</td>
</tr>
<tr>
<td>Computer Studies Major</td>
<td>.70713</td>
<td>.12297</td>
<td>.26571</td>
<td>5.750</td>
<td>.0000</td>
</tr>
<tr>
<td>Acceptance of the Behaviour</td>
<td>.14932</td>
<td>.04942</td>
<td>.08645</td>
<td>2.004</td>
<td>.0457</td>
</tr>
<tr>
<td>Access to Legal Software</td>
<td>.34934</td>
<td>.04992</td>
<td>.26623</td>
<td>6.998</td>
<td>.0000</td>
</tr>
<tr>
<td>OK if Not Making Money</td>
<td>.11774</td>
<td>.04765</td>
<td>.09042</td>
<td>2.455</td>
<td>.0144</td>
</tr>
<tr>
<td>Concern for Difficulty and Cost</td>
<td>.15878</td>
<td>.04924</td>
<td>.11462</td>
<td>-3.213</td>
<td>.0014</td>
</tr>
</tbody>
</table>

### Table 9: How Often Does a Student Purchase Illegal Software

<table>
<thead>
<tr>
<th>Variables Selected (PIN = .05)</th>
<th>B</th>
<th>SE</th>
<th>Beta</th>
<th>T</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td>-.17785</td>
<td>.08703</td>
<td>-.09918</td>
<td>-2.030</td>
<td>.0430</td>
</tr>
<tr>
<td>Family Background</td>
<td>-.33950</td>
<td>.07071</td>
<td>-.05977</td>
<td>-1.973</td>
<td>.0491</td>
</tr>
<tr>
<td>Own PC</td>
<td>.37424</td>
<td>.19875</td>
<td>.18183</td>
<td>3.790</td>
<td>.0002</td>
</tr>
<tr>
<td>Computer Studies Major</td>
<td>.19051</td>
<td>.09510</td>
<td>.10006</td>
<td>2.098</td>
<td>.0365</td>
</tr>
<tr>
<td>Acceptance of the Behaviour</td>
<td>.19631</td>
<td>.04326</td>
<td>.19802</td>
<td>4.538</td>
<td>.0000</td>
</tr>
<tr>
<td>Concern for Difficulty and Costs</td>
<td>-.11973</td>
<td>.04349</td>
<td>-.11858</td>
<td>-2.712</td>
<td>.0069</td>
</tr>
</tbody>
</table>

---

**THE AUSTRALIAN COMPUTER JOURNAL, VOL. 22, NO. 4, NOVEMBER 1990**
likely use copied software, but Social Science majors would not. As for students' attitudes, those who felt that copying software was acceptable behaviour, those who felt that they had difficulty in accessing legal software, and those who believed that copying software was permissible if not used to make money, would more likely use copied software. However, students' attitudes towards publishers, their attitudes towards property rights, and their concern for difficulty and cost in copying software, had no effect on whether they actually use copied software or not.

Allowing Others to Copy One's Software: The regression result for this dependent variable was almost identical to the previous one. Since these two variables were highly correlated, this seems reasonable. The only difference between them was that Concern for Difficulty And Cost in Copying Software was significant in this case but not in the other. The negative B suggests that those more concerned about difficulty and cost associated with copying software would be less likely to allow others to copy software. This may be because when students used copied software, the software was already copied, so their concern for costs and difficulty did not affect their behaviour. However, when they had to allow others to make copies, or when they had to make copies for others, the issue became more relevant.

Buying Illegally Copied Software: The buying of illegally copied software involved very different behaviour. Students from a lower family background, students who owned PCs, male students; and computer studies majors, would more likely buy illegally copied software. Also, those who felt that copying software was acceptable, and those who were less concerned about difficulty and cost associated with copying software, would more likely buy illegally copied software.

Effects of the independent variables on the three dependent variables are summarized below.

6. DISCUSSION
Results from this study have highlighted several interesting points. Only about 10% of the students said that they had never used copied software, or had never allowed others to copy their software. This means that the number of students copying and using copied software was higher than that from the two studies conducted by Shim and Taylor and Lin. However, because the population studied was somewhat different, it is unfair to compare the results directly. In general, it is reasonable to find more copying among students because of their higher level of curiosity. Results from this study simply quantify the seriousness of the problem.

More important is the large number of students that had purchased illegally copied software. Only 24% said that they had never purchased illegally copied software (another 32% said that they rarely purchased illegal software, but still about 45% said that they purchased illegal software sometimes; frequently; or all the time). This strong demand will make stopping the sales of illegally copied software more difficult. It also reveals an important issue: if students were willing to pay for the software, why didn’t they buy the original one?

One factor identified in the factor analysis may provide some indications about this question. A clear majority of the students felt that original software was not readily available. Almost all (90%) felt that original software was too expensive; and a clear majority (78%) said that it was more convenient to copy software than to buy the original.

Software distribution channels in Hong Kong may have a lot to do with this situation. In the U.S., heavy competition in the software retail market makes buying original software very easy. PC software is sold everywhere: in software chain stores such as Egghead Software; in Record companies such as The Warehouse, and through large mail-order outlets such as PC-Connection. Also, original software is sold at discounts up to 70% of the list price. Students sometimes can even buy software at a lower price. For example, Borland International, publisher of many well known software products, including Turbo Pascal, Turbo C, SideKick etc., has an educational discount program, whereby students and faculty
members can buy Borland products at prices that are even lower than mail-order. After-sales technical support is provided by the publisher, mainly through toll-free phone numbers. However, software in Hong Kong is typically sold at computer stores at about 5% above the list price, which means the same software may be 1.5 to 3 times more expensive than in the U.S. For example, Wordstar costs about US $180 in the U.S. and US $300 in Hong Kong. Grammatik IV, a popular grammar checker, costs about US $120 in a computer store in Hong Kong. The same program can be bought for about US $50 in the U.S.. In addition to the price difference, finding a retailer that carries a not-so popular software product in Hong Kong can be quite difficult.

This may create a vicious circle for the software industry in Hong Kong. Because users are copying and buying illegally copied software, sales for original software are weak, and distributors have to charge a high price to cover the overheads (after-sales support is usually provided by the local distributor, not the software publisher), and only carry popular products. However, because original software is expensive and difficult to find, users, especially students who cannot afford to pay for expensive software, have to get them through other means. To reduce copying among students, publishers and distributors may have to take drastic action. It may be very helpful if marketing programs similar to the one offered by Borland can be introduced in Hong Kong (Ashton-Tate announced a similar program in August 90). Alternatively, distributors may price the software and support separately so that those who do not need much support need not pay a premium price. In addition, it may be profitable to adopt warehouse-style software retailing, where software is sold at discount price with little support provided.

The factor analysis and the regression analysis have also revealed several interesting points. One is students' attitude towards publishers. It is unclear why a significant proportion of students had a negative attitude towards software publishers. On the other hand, this attitude did not affect their behaviour. The implication of this finding needs further investigation.

As regards the attitude towards property rights, most students said that one should pay for goods and services received and it is unfair to the author to copy software. However, less than one-third said that it is unethical to copy software; or that copying software will hurt everyone in the long run. It appears that students were treating software very differently from other types of products or services. Perhaps this is because when software is copied, nothing tangible is "lost" or "stolen". On the other hand, this attitude did not appear to affect the actual copying or using of copied software. This poses a question as to whether education of computer ethics will be effective in reducing the copying of software.

Alternatively, making software copying a socially less acceptable behaviour and increasing the cost and difficulty associated with it, may be more effective in combating the problem since both factors affected all three dependent variables significantly. This is consistent with the findings of Straub (1987) that security measures were effective in reducing computer abuse. Some companies already have regular inspection of their employees' PCs to see whether any unauthorized software is being used. Schools could implement similar programs. Similarly, it would be possible to patrol computer rooms of schools, with students found using software not supplied by the school being required to prove that it is an original copy, or face disciplinary action. This would increase the cost and difficulty of using unauthorized software, and send a strong signal to the students that the behaviour is not acceptable.

CONCLUSION

This study has accomplished several objectives. It provides policy makers with statistics on the seriousness of the problem among students. It also identifies factors that significantly affect students' behaviour, which may help to develop better strategies to combat the problem. The results suggest that better marketing of original software and increasing the penalties associated with copying software and with using copied software may be more effective in controlling the problem than education in computer ethics.

Although students are a major group of computer users in Hong Kong, there are other large groups of users that cannot be ignored, and many software publishers are focusing their attention on corporate copying of software. Similar studies should be conducted among these users to understand why they copy software and what factors affect their behaviour. One result from this study was that Employment Status had no effect on any of the three dependent variables, which suggests that results from this study may be applicable to a larger population.

REFERENCE


APPENDIX: THE QUESTIONNAIRE USED

SECTION 1: PERSONAL BACKGROUND
(1) Age: ______
(2) Sex: Male O Female O
(3) Years of full-time (or equivalent) education: ______ 2 years part-time equals 1 year full-time
(4) CPHK Department:
Accountancy O Computer Studies O
Applied Linguistics O Economics & Finances O
Applied Mathematics O Electronic Engineering O
Applied Science O English O
Applied Social Studies O Law O
Building & Construction O Manufacturing Eng. O
Business & Mgmt. O Public & Soc. Admin. O
(5) Are you working? full-time O part-time O no O
If yes, (5a) What is your annual income? $ ______
(5b) Nature of job:
labor intense O technical O
clerical O managerial O
service O professional O
teaching O other ______
(5c) How many hours per week do you work? ______
(6) Money available to you this year (including income/grant/loan/family support etc.) $ ______
(7) Religion: Buddhism O Traditional Chinese O
Catholic O None O
Protestant O Other ______
(8) Where did you receive the majority of your primary education?
Hong Kong O Malaysia O North America O
China O S.E. Asia O Australia O
Taiwan O Europe O Other ______
(9) What is your father’s occupation?
labor intense O technical O
clerical service O managerial O
professional O
other ______
(10) What is your mother’s occupation?
labor intense O technical O
clerical/service O managerial O
professional O
other ______
(11) How would you classify your family background?
greencollar O upper-middle class O
middle class O elite O
(12) Do you own a personal computer? Yes O No O
(13) How would you describe your own computer knowledge?
know nothing O average knowledge O excellent knowledge O
know very little O good knowledge O
know something O slightly knowledge O
(14) How often do you use personal computers for school or personal work?
ever O rarely O sometimes O frequently O
(15) Have you used software that are copied without proper authorization?
never O rarely O sometimes O frequently O
(16) Have you allowed someone to copy your software?
never O rarely O sometimes O frequently O
(17) Have you made unauthorized recording of music and/or video?
never O rarely O sometimes O frequently O
(18) Have you copied textbooks without permission?
never O rarely O sometimes O frequently O
(19) Have you copied homework assignments from your classmates?
never O rarely O sometimes O frequently O
(20) Have you purchased software that is copied illegally?
never O rarely O sometimes O frequently O
(21) Have you purchased or used counterfeit brand-name products, such as handbags, T-shirts etc.?
never O rarely O sometimes O frequently O

For questions 14-21, please indicate how often you perform the activities mentioned.

<table>
<thead>
<tr>
<th>never</th>
<th>rarely</th>
<th>sometimes</th>
<th>frequently</th>
<th>all the time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SECTION 2: PLEASE INDICATE WHETHER YOU AGREE OR DISAGREE WITH THE FOLLOWING STATEMENTS

<table>
<thead>
<tr>
<th>strongly disagree</th>
<th>disagree</th>
<th>neutral</th>
<th>agree</th>
<th>strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (1) Original software is too expensive.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 (2) It is more convenient to make unauthorized copies than to purchase the original one.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 (3) It is unethical to use software without paying for it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 (4) Copying software is O.K. because there is no penalty associated with doing so.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 (5) Copying software will hurt everybody in the long run.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 (6) If I can get something free of charge, I will not pay for it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 (7) If software programs cost less, I will not use unauthorized copies.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 (8) Copying a copy-protected software is challenging and fun.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 (9) I will not copy software if there is penalty associated with doing so.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 (10) I will not use unauthorized copies of software if more money is available to me.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 (11) I use unauthorized copies of software because my friends are doing the same thing.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 (12) Software publishers are making too much money.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 (13) One should pay for goods and services received.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 (14) I copy software because it is easy to do.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 (15) Software publishers don’t want unauthorized copying because they are greedy.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 (16) I will not attempt to copy a software if it is heavily copy protected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 (17) If I write a program, I don’t mind others copying it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 (18) One should pay for a software that he/she uses it to make money.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 (19) Software users should not have to worry about the integrity of software publishers.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 (20) Not having the original manual and vendor support creates a problem for users of copied software.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 (21) It is O.K. to copy software as long as I do not use it for commercial purposes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SOFTWARE COPYING AMONG STUDENTS

(22) It is O.K. to use copied software if you purchase it from a dealer.
(23) It's ashamed if others know that I use copied software.
(24) It is a violation of copyright laws to copy software.
(25) Somebody spent a great effort to produce a software, therefore it is not fair to copy it.
(26) Services provided by software publishers, such as support hot line, newsletter, program update etc., justifies buying an original software.
(27) If software publishers don't want their products copied, they should make them copy-proof.
(28) Copying a software will actually make it more popular, hence benefits the publisher.
(29) Software copying tools exist implies copying software is O.K.
(30) A good computer user should have a large library of software.
(31) Software publishers spend a lot of effort in packaging and marketing the software, the program itself actually does not worth that much.
(32) I copy a software because I just want to learn it.
(33) Regardless of the cost and under all circumstances, I'll copy a software and not purchase the original one.
(34) Students should be allowed to copy software for completion of their course work and assignments.
(35) The software packages available at school are outdated when compare to the outsider market.

BIOGRAPHICAL NOTES

Gordon Wong is a lecturer in the Department of Information Systems, City Polytechnic of Hong Kong. Dr Wong received his Ph.D. in Computer Information Systems from the Univ. of California, Los Angeles. His main research interests are in Software Piracy and Information Technology Assimilation.

Andy Kong is also a lecturer in the Department of Information Systems, City Polytechnic, Hong Kong. Mr Kong received his B.Sc. degree from the Univ. of Ulster, U.K., and a M.Eng. from the Univ. of Wales Institute of Science and Technology. He is currently a doctoral student at the Department of Business & Management, Univ. of Hong Kong.

Sinto Ngai is a lecturer in the Department of Computer Science, City Polytechnic of Hong Kong. Mr Ngai is a graduate of the Univ. of California, Los Angeles, where he received a B.S. in Mathematics and Computer Science. He also received a M.S. in Computer Science from the Univ. of Southern California.

BOOK REVIEWS

NEW REVIEWERS REQUIRED and UPDATE OF RECORDS

Over recent weeks, I have been streamlining the operation of the Book Review Section of the Journal. This has involved culling our records, and updating the particulars that are held on potential reviewers. As can be imagined, this is rather a large task.

All those currently in the records have been written to with a request that the particulars held on file be updated. Each person who has reviewed books for the Journal should have received such a request. A reminder, then, to return this form as soon as possible.

I'm sure, however, that there are other readers who would be willing to perform the task of reviewing books for the Journal. If this is the case, then please return the form found elsewhere in this issue. Referees are also required for submitted papers, and the same form can be used to register for this task, too. If perchance you have reviewed books for us previously and have not received the form in the separate mailing, please return the form in this issue and annotate it appropriately to let me know that your particular record needs further checking.

If there are others with whom you have contact who are not members of ACS, but who may be willing to review books for us (for example, I have a practising psychiatrist in mind to review a book on ‘Computer Addiction’), there are two things you could do. Firstly encourage them to consider joining ACS, and secondly photocopy and pass on the form for them to return. This will be particularly applicable to large, and possibly specialised firms, and also to academic departments.

The book review section is a very important part of the journal, and your help is earnestly sought in helping to maintain its high standard.

Chris. Andrews
Associate Editor
Book Reviews
The lessons of recent EC and US developments for protection of computer software under Australian law

Peter Waters
Associate, Gilbert & Tobin, Technology Lawyers

Peter G. Leonard
Partner, Gilbert & Tobin, Technology Lawyers

The law needs to provide sufficient protection for computer software if programmers are to have the incentive to invest the considerable creative effort and resources required to develop new programs. On the other hand, advances in the software industry are incremental, with new innovations building on past developments. This stepping stone progress in software development could be stifled if the legal protection of software was too extensive. Copyright law, which applies to computer software, endeavours to strike a balance between these competing policy goals by protecting the particular expression of an idea but not protecting the idea itself. This article examines the difficulties in applying this abstract dichotomy to computer software. The Copyright Law Review Committee is currently reviewing the legal protection of software. This article considers recent overseas developments in the legal protection of software, particularly the European Community's proposed Directive and recent US court cases.

Keywords and phrases: Copyright, literal copying, nonliteral copying, user right to adapt, EC law, US law.

CR categories: K.1, K.4, K.5.1.

1. INTRODUCTION

In the past there has been much debate about whether a software developer's rights are best protected by contract, patent or copyright law, or by a new specially developed branch of the law. The scepticism which was widely expressed about the suitability of copyright protection for software has been replaced with a general acceptance at the international level that copyright is the way to go (CEC, 1988: WIPO, 1983). Debate has shifted to how to tailor the traditional notions of copyright to the specific features of computer software (CLRC, 1990), and the current Enquiry by the Copyright Law Review Committee is mainly in this vein. It is interesting to compare the CLRC's Issues Paper with recent trends in the US and the European Community's Directive on the Legal Protection of Computer Software (CEC, 1989), which has become a lightning-rod for much of the international debate.

2. SCOPE OF COPYRIGHT PROTECTION

Copyright infringement is readily established where a software program has been copied in its entirety and is used verbatim. However, it is a more tricky exercise where the allegation is that copyright has been infringed by a programmer "borrowing" elements from other software in the writing of his or her own program.

Software is not a monolithic block of undifferentiated code, but is a complex construction of modules, routines and sub-routines, purposely arranged in a particular sequence and which the user drives through a menu of commands. The allegedly infringing programmer may have copied literally some lines or blocks of code from the earlier program, and set that code amongst lines of code of his or her own creation. The copied code may represent the heart of the earlier program, and the added code may only be to camouflage the extent of the literal copying. Conversely, the copied code may be utility elements or building blocks or may simply be included in order to achieve interoperability between the two programs. However, the later program might not even contain any lines of duplicated code, but the allegedly infringing programmer may have borrowed heavily from the structure of the earlier software, by replicating, for example, the way in which the program requests, stores, retrieves and analyses data; the way the program divides tasks between multiple modules and sub-routines; or its command menu. This is known as "non-literal" copying.

Copyright law traditionally draws a distinction between ideas behind a protected work, which are not protected, and the author's expression of those ideas, which are protected. This ensures an individual's right to profit from the product of his or her intellectual efforts while also enriching the store of ideas which is available to all. While the ideas/expression dichotomy is readily stated and understood in the abstract, it is difficult to determine where exactly to draw this line through the complex hie-
rarchy of programming elements which make up a computer program. As one commentator stated:

Drawing the line too liberally in favour of copyright protection would bestow strong monopolies of specific applications upon the first to write programs performing those applications and would thereby inhibit other creators from developing improved products. Drawing the line too conservatively would allow programmers' efforts to be copied easily, thus discouraging the creation of all but modest incremental advances (Menell, 1988, pp. 1047-1048).

(a) EC Approach
The European Communities Commission stated in the draft Directive that copyright protection should be available to the "program as a whole and to its constituent parts insofar as they represent a sufficient degree of creativity to qualify as 'works' themselves" (CEC, 1989, C91/9). Article 1.3 of the draft Directive states:

Protection in accordance with this Directive shall apply to the expression in any form of a computer program but shall not extend to the ideas, principles, logic, algorithms or programming languages underlying the program. Where the specification of interfaces constitutes ideas and principles which underlie the program, those ideas and principles are not copyrightable subject matter.

The EC Commission thought that the moves towards standardisation and interoperability should be encouraged in the computer industry, noting that:

In order to produce interoperative systems it is necessary to replicate the ideas, rules or principles by which interfaces between systems are specified, but not necessary to reproduce the code which implements them. Ideas, rules or principles are not copyrightable subject matter. Such ideas, rules or principles may be used by any programmer in the creation of an independent implementation of them in an interoperative program (CEC, 1989, C91/9 para 3.11).

The EC Commission thought that algorithms were "equivalent of the words by which the poet or the novelist creates his work of literature, or the brush strokes or musical scales of the composer" (CEC, C91/5 para 2.4). The modules, routines and sub-routines which were built on these algorithms "may all qualify for protection independently of the protection given to the program as a whole as a compilation of such elements," (CEC, C91/9 para 3) provided those compilations were sufficiently original. However, the EC Commission also noted that "many sub-routines are [so] commonplace in the industry that they may have been placed or have fallen into the public domain or they may be de facto standard routines" (CEC, C91/9 para 4(a)).

The EC Commission did not specifically address the issue of whether the "non-literal" aspects of the program, such as the arrangement of the modules and sub-routines, were copyrightable. However, it seems to have assumed that the organisation and structure of a program would fall within the category of "logic and principles...underlying the program", and would not be subject to copyright protection.

The EC's Economic and Social Committee and the European Parliament have recommended that Article 1.3 of the proposed Directive be scaled back to simply restate the general distinction between ideas and principles, and that the statement that "logic, algorithms or programming languages" are unprotectable ideas rather than protectable expressions should be deleted. The Committee and the Parliament did not necessarily disagree with the Commission's categorisation of these programming elements and ideas, but thought that:

logic and algorithms are not clear terms and are frequently interchangeable. Few would argue that logic, algorithms and programming languages are [copyrightable],...[and] specific reference...to these terms as requiring exclusion from protection only serves to confuse and cause unnecessary debate (ESC, 1989, C329/5 para 3.3.2.2).

(b) US Approach
Recent developments in the United States suggest that the categorisation of programming elements as either ideas or expression may not be as straightforward or uncontroversial as the EC Economic and Social Committee would have, particularly in relation to the "non-literal" aspects of a computer program.

The US Copyright Act takes a similar approach to the EC Commission's version of the Directive, providing in section 102(b) that:

In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle or discovery, regardless of form in which it is described, explained, illustrated or embodied in such work.

Echoing the EC Commission, the House of Representatives Report adopting this provision noted that:

Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in the computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law (Gervaise Davis, 1990).

The US Copyright Office has also stated that "copyright protection is not available for ideas, program logic, algorithms, systems, methods, concepts or layouts" (17 USC, 1982). An early computing case, Synercom v University (1978) confirmed the proposition that the sequencing and arrangement of data was not copyrightable. University Computing Co closely studied Synercom's computer program and then developed its own program which had exactly the same sequencing of input on punched cards. The Federal District Court held that the sequence and ordering of the input was the idea around which Synercom's program was built, and not an expression. As University developed its own program around the same format, it had created "original expressions of the copied idea".
However, US courts more recently have held that “copyright protection is not limited to the literal aspects of a computer program, but rather ... it extends to the overall structure of a program” (Broderbund v Unison, 1986, p. 1133). In Whelan v Jaslow (1985) the defendant contracted with a software house for software to manage the operations of its dental laboratory. Realising the potential market for the software among smaller dental laboratories, the defendant rewrote the software in a different language suitable for personal computers. The final product contained one line of code which was identical to a line of code in the original program, but the defendant had closely studied and replicated the manner in which information flowed from one function to another. The court defined the expression of the idea in a computer program as “the manner in which the program operates, controls and regulates the computer in receiving, assembling, calculating, retaining, correlating, and producing useful information either on a screen, print-out or by audio-communication”. The unprotected idea was no more than “the mere idea or concept of a computerised program for operating a dental laboratory” (Whelan v Jaslow, 1985, p. 1320).

If the unprotected idea is defined in this limited and generalised fashion, then it necessarily follows that almost every element of the program implementing that idea will be copyrightable expression. The extraordinary depth of protection which flows from this approach is well illustrated by the recent decision in Lotus v Paperback (1990). Paperback had designed its spreadsheet program to work like the Lotus 1-2-3 program “keystroke for keystroke” and with the same “command tree”. Paperback’s admitted purpose was to allow users to transfer spreadsheets created in Lotus 1-2-3 to VP-Planner without the loss of functionality or familiarity for the user. This could only be achieved if all menu commands had the same first letter and were in the same place in the menu hierarchy as in Lotus 1-2-3. Paperback had written an entirely different code for its program, so that what had been copied was the “non-literal” arrangement of the user interface and the command sequencing. Paperback argued that it had done no more than replicate ideas, or expressions that were so utilitarian or commonplace in the industry that they should be unprotectable.

The Federal District Court held that the basic unprotected idea underlying the two programs was that of an electronic spreadsheet, and that other software developers had managed to express the same idea in ways which were very different to Lotus 1-2-3. Keeton J agreed that certain common elements between the two programs were also unprotectable because there are limited alternative methods for making a spreadsheet; for example, the use of a two-line moving cursor. The use of the “/” key to invoke the menu command was also permissible simply because there were not many other keys which were readily availa-

ble to perform that function.

However, beyond these peripheral programming elements, his Honour held that the overall structure, the order of commands in each menu line, and the choice of letters or words to represent commands, were all “capable of being expressed in many if not an unlimited number of ways, and that the command structure of Lotus 1-2-3 is an original and non-obvious way of expressing a command structure” (1990, p. 68). The Judge acknowledged that some of these elements were commonplace in other programs, but when combined in a unique command structure the whole was a protectable expression.

Keeton J noted that Paperback could have offered the unique capabilities and additional functions of its software as an “add-in” to the Lotus program, and that Lotus had released a “Lotus Developer Tools” to help developers “legitimately” write programs that work with Lotus 1-2-3. Alternately, Paperback could have included a help function in its software which showed users the equivalent commands between the two programs. Displaying a strong disapproval of “cloning”, a disapproval which apparently drove much of his reasoning, his Honour concluded:

By instead selling a stand alone product that completely replaces 1-2-3. [Paperback] have not merely sold and profited from only their incremental additional expression. Rather, they seek permission to profit also from copying Lotus’ protected expression (1990, p. 79).

His Honour added that “the desire to achieve compatibility or standardisation cannot override the rights of authors to a limited monopoly in the expression embodied in their intellectual work” (1990, p. 60).

Paperback also argued that the macro commands were not part of the underlying program performing the functions selected by the user, but rather were a convenient and utilitarian method by which the user communicates with and drives the program like keys on a typewriter (17 USC s. 101: Williams v Bally, 1983). These commands, such as “/F” for “File”, were, Paperback asserted the very “procedure, process system or method of operation” which section 102(b) of the US Copyright Act characterises as an unprotected idea. Keeton J’s consideration of these arguments was cursory and very unsatisfactory. He noted that it should not be assumed that a computer language around which a program is constructed and operates is not separately copyrightable.

The lesson of the Lotus decision is that programmers should not mimic common elements of a successful program, even if it has become the de facto industry standard. If others have approached the same problem in a different manner, the newcomer may need also to take a different approach. If there is only one other product in the market, the new start-up should be prepared to show that any similarities are strictly mandated by functional considerations, and that it has exhausted other means of achieving the same functionality. The reasoning of the Lotus court
would appear to travel beyond the outer limits of the copyright protection envisaged by the EC in its proposed directive (Khatcherian, 1990, p. 18).

(c) The Australian Position
The definition of computer program in the Australian Copyright Act does not give much indication of the "depth" of copyright protection afforded under Australian law. The Act does not contain an exclusion of procedures, modes of operation or logic paralleling the US Copyright Act or the EC Commission's draft Directive.

Section 10 of the Act defines a computer program as "an expression...of a set of instructions intended...to cause a device having digital information processing capabilities to perform a particular function". In a familiar cant, the Explanatory Memorandum said of this definition:

The phrase "expression...of a set of instructions" is intended to make clear that it is not an abstract idea, algorithm or mathematical principle which is protected, but rather a particular expression of that abstraction. The word "set" indicates that the instructions are related to one another rather than being a mere collection (Dyason v Autodesk, 1990, pp. 20-21).

The recent decisions in the Autodesk litigation shed some oblique light on where this definition pitches the idea/expression dichotomy in the hierarchy of programming elements. Autodesk, an American software house, licensed an architectural software package which could only be used in conjunction with a hardwired backing device (AutoCAD). A local engineer, by using an oscilloscope applied to the locking device, discovered how the lock altered the input from the computer program, which on transmission back to the program "unlocked" it. He developed his own device to produce the same altered pattern ("Auto-key"), but other than replicating the function of the Autodesk lock, the internal workings of the two devices were completely different. In other words, there was no literal copying.

In the lower court decision, Northrop J held that the definition of computer program in the Act raised the function of the two programs to a pre-eminent position in determining whether one program was substantially similar to the other, and therefore is a copy of the other.

Protecting the function of the computer program extinguishes the ideas/expression dichotomy, or very nearly so. Northrop J's approach provides a level of protection somewhere approaching that of patents, which bestow a monopoly in ideas.

The three appeal judges rejected Northrop J's approach of measuring substantial similarity based on functionality, and substituted a test more closely constructed around a comparison of the literal binary code.

Lockhart J commented that the definition of computer program:

demonstrates that it is the expression of the requisite set of instructions that constitutes the program. This establishes, consistently with accepted thinking in copyright law, that the Act intends copyright protection, not on abstract ideas or mathematical principles, but on the particular expression of the ideas or principles in the form of sets of instructions (p. 15).

Beaumont J thought that the "set of instructions" which constituted the protectable element of the computer program was the "orders or directions, used together, being numbers or symbols which cause a computer to perform some specified action" (p. 25).

Echoing the conclusions of the other two judges, Lockhart J concluded:

The recent decisions in the Autodesk litigation shed some oblique light on where this definition pitches the idea/expression dichotomy in the hierarchy of programming elements. Autodesk, an American software house, licensed an architectural software package which could only be used in conjunction with a hardwired backing device (AutoCAD). A local engineer, by using an oscilloscope applied to the locking device, discovered how the lock altered the input from the computer program, which on transmission back to the program "unlocked" it. He developed his own device to produce the same altered pattern ("Auto-key"), but other than replicating the function of the Autodesk lock, the internal workings of the two devices were completely different. In other words, there was no literal copying.

In the lower court decision, Northrop J held that the definition of computer program in the Act raised the function of the two programs to a pre-eminent position in determining whether one program was substantially similar to the other, and therefore is a copy of the other.

Protecting the function of the computer program extinguishes the ideas/expression dichotomy, or very nearly so. Northrop J's approach provides a level of protection somewhere approaching that of patents, which bestow a monopoly in ideas.

The three appeal judges rejected Northrop J's approach of measuring substantial similarity based on functionality, and substituted a test more closely constructed around a comparison of the literal binary code.

Lockhart J commented that the definition of computer program:

demonstrates that it is the expression of the requisite set of instructions that constitutes the program. This establishes, consistently with accepted thinking in copyright law, that the Act intends copyright protection, not on abstract ideas or mathematical principles, but on the particular expression of the ideas or principles in the form of sets of instructions (p. 15).

Beaumont J thought that the "set of instructions" which constituted the protectable element of the computer program was the "orders or directions, used together, being numbers or symbols which cause a computer to perform some specified action" (p. 25).

Echoing the conclusions of the other two judges, Lockhart J concluded:

The algorithm (that is the procedures for solving the particular problem in a finite number of steps) employed by each [lock] are different and the implementation of each is different. There is not a shred of similarity between the two programs or sets of instructions (p. 15).

Autodesk had also argued that if the Auto-key was not a literal copy, then it was at least a "version" of the AutoCAD lock, which would still offend Autodesk's copyright. However, the three judges held that for one thing to be a version of another it had to share a greater degree of similarity than function.

Lockhart J stated that:

It is not in my view necessary for there to be an adaptation of a copyright work that the impugned version of the work embodies or preserves the original intrinsic qualities of the copyright work. There are indicia within the definition of "adaptation" itself which demonstrate that a work may be an adaptation of another work notwithstanding considerable differences between the two works. Indeed, the word "version", meaning a variant of something, necessarily requires that there be change or variation. Though it goes too far to say that something is a version of another thing if the two are essentially different (p. 18).

The appeal court's discussion of the phrase "expression of a set of instructions" in terms of the binary code clearly suggests that only the literal aspects of the program will be copyrightable (17 USC s. 101). On the other hand, the three judges thought that the concept version had to be given a wide construction, and as Lockhard J apparently suggested in the passage quoted above, that there need not be literal elements in common between the two programs. This is an invitation for arguing the US non-literal copyright and look and feel cases in Australia. However, Autodesk would not have succeeded under US non-literal copying doctrine, even as applied at the most broad in the Lotus case. The replicated function was reached by an entirely different programming pathway, literally and non-literally.

The analogy of Anglo-Australian cases on the protection afforded plots and other elements of literary works also suggests extensive protection for the literal and non-literal aspects of software. In Rees v Melville (1911-16) Eady LJ remarked that in order to constitute an infringement it was not necessary to copy the words of the original dialogue, but the situations, incidents, and way in which ideas were worked out and presented could compromise a
material part of a play which is protected by copyright (1911-16, p. 174). Pickford LJ commented that copyright would be infringed by the copying of the idea and the arrangement of the incidents, taking into account the extent to which both plays employed stock incidents. Like playwright, the programmer arranges a myriad of programming elements, some of them quite commonplace, into a unitary and carefully wrought structure (pp. 174-175: contrast *Kelly v. Cinema*, 1928-35).

(d) Implications for Reform

It would be a mistake to view ideas and their expression as separate discrete categories, and to imagine that it is possible to forensically determine whether a programming element belongs in one category or the other. Rather, the distinction is better conceived as a continuum which runs from a pure intellectual concept through to its literal reduction to a code of “0s” and “1s”. In between is a grey area of programming elements which are part ideas, part expression, or part commonly used programming vocabulary (Root, 1983, p. 1299). Where the line is drawn in that grey area is a policy and not a legal question. A balance has to be struck between an individual developer’s rights to commercially exploit his or her intellectual product, and the social values in the circulation of new developments and the achievement of interoperability of computers.

The CLRC has canvassed the introduction of protection for software against non-literal copying, along the lines of the US test. Substantial arguments can be marshalled in favour of non-literal copyright. A program’s efficiency turns on the arrangement of its modules, and sub-routines, and although two programs may have the same functionality, one may be much more commercially successful than the other because it has a cleverer structure. A greater investment may be made in devising a rigorous mathematical process or algorithm than in the literal translation into code (Keplinger, 1981, pp. 486-487; *Whelan v. Jaslow*, 1985, p. 1230). Equally, as Keeton J said of Lotus 1-2-3, the clever arrangement of the user interface can be the key to higher user acceptance (*Lotus v. Paperback*, 1990, p. 63). However, there are persuasive arguments that non-literal copying would tip the balance too far towards protection of software and have a chilling effect on development within the industry.

First, the “plot” cases are an unhelpful analogy in determining the extent of protection for the non-literal aspects of software. An infringer of copyright in a book can readily discover the essential elements of the work by simply reading it, and then easily produce a non-literal version by shuffling words and phrases in a book while retaining the same sense as the copied work. The considerable intellectual and economic effort that the author puts into the work can be undermined by disproportionately minimal effort on the part of a non-literal infringer. The courts therefore have good reason to be vigilant in preventing the copying of creative elements of the larger literary work.

The dynamic of non-literal copying, and the relative balance of effort between the original author and the “borrowing party” is entirely different in the case of software. As one commentator has said:

Computer programs involve exact instructions with each statement precisely linking with other statements to produce a functional program. While one could change the non-critical parts of source code programs, such as the numbers for each program statement or the names of any variables, changing the operative statements would require precise replacements and the investment of substantial knowledge and time by the pirate. One cannot simply “approximate” the entire copyrighted program and create a similar operative program without the expenditure of almost the same amount of time as the original programmer expended (Root, 1983, pp. 1289-1290).

In other words, unlike other literary works, the replication of non-literal aspects between different programs is less likely to be a case of disguised copying, than simply a tangible mark of the influence which the solutions reached by the earlier programmer have had in the separate intellectual effort undertaken by the later programmer.

Second, the software industry moves forward in incremental steps, with new innovations building on past developments. While there are a few icons of the industry who conceive completely new ways of writing software, such as Dr Wang or Steve Jobs, most programmers study, mimic, adapt and improve upon the work of these pathfinders. Bluntly put, this “stepping-stone” progress in computer programs involves plagiarising in some manner the underlying copyrighted work (Root, 1983, p. 1292).

Third, software is not valued for its own intrinsic worth, but for its ability to do a defined task faster, more cleverly and with fewer errors than the versions that came before it. While there may be an infinite number of variations to the “boy-meets-girl” saga, there may be a finite number of ways in which routines can be written to perform a particular task (*Goldstein*, 1986, pp. 1122-1123). Protecting software structure, sequencing, commands and language could rapidly exhaust those alternatives, or require resort to more convoluted and inefficient structures simply to achieve the required level of originality.

Finally, with literary works, variety of expression of ideas has an independent social value. The opposite is the case with software: in order to achieve interoperability between different makes of software, many sharing of language, commands, structure and code is required. Overly strict application of copyright could mean users are confronted with a Tower of Babel of incompatible systems.

Protection of the non-literal elements of a program would be particularly unsuited to the infant Australian software industry. While the Australian industry is undoubtedly vigorous and creative, the reality is that most significant developments occur overseas and many Aus-
3. REVERSE ENGINEERING

Reverse engineering involves the decompilation of software to discover how it works, and then the use of that discovered knowledge to create competing software. The reverse engineering issue arises whether or not the non-literal aspects of a program are copyrightable, but where the category of protected elements is extended, much less of the discovered knowledge from the decompiled software could safely be used (Khatcherian, 1990, p. 24). The reverse engineering debate is made more complex and confusing by the technological fact that the running of a program, whether by an ordinary user, or a reverse engineering party, requires its copying into the computer's memory (Dyason v. Autodesk, 1990, Sheppard and Beaumont JJ).

Much of the debate over the proposed EC Directive has focused on the vexed reverse engineering issue. The European Council for Interoperative Systems (“ECIS”), composed of systems manufacturers such as Unisys, Sun Microsystems, Amdahl, Bull, Olivetti and Fujitsu, lobbied for inclusion in the Directive of a specific permission to reverse engineer software. This is vigorously opposed by the Software Action Group for Europe (“SAGE”), which includes IBM, Digital, Microsoft and Lotus.

ECIS says that reverse engineering should be permitted to ensure free access to the non-protected ideas and principles which underlie a computer program. With books and records, the unprotected idea is readily discerned by reading or listening, but the ideas behind a computer program can only be discovered by pulling it apart, much as the workings of a machine will only emerge on disassembly. ECIS asserts that prohibiting reverse engineering would effectively give a programmer a monopoly in ideas (Khatcherian, 1990, pp. 2-3).

SAGE counters that reverse engineering is not strictly necessary to access the ideas underlying a program, but that the ideas will be discoverable from examining the documentation released with the software, running test data through the program, and even reading the object code, which, while difficult, can be done by skilled programmers (Khatcherian, 1990: McGrady, 1983). SAGE asserts that the more sinister purpose of reverse engineering is to permit “cloners” to piggyback on the considerable effort and investment put into the development of software by industry leaders:

the public’s need for access to the copyrighted work is fully satisfied by the copyright owners marketing the original. A competitor who reverse-engineers a copyrighted computer program is not at all interested in increasing that access: to the contrary, his only purpose is to get the public to purchase his work rather than the original, thereby eliminating the market for the original (Party, 1985, p. 401).

It is claimed that the main beneficiaries of reverse engineering would be “those countries, such as Japan, which, although major producers of hardware, lag behind the United States and Europe in the production of software, which they are naturally keen to control” (CLACR, 1989, pp. 11-12), and that these companies are now seeking to “overcome their commercial failures by claiming a right to engage in reverse engineering” (BSA, 1989, p. 13).

Even if this purpose motivated the reverse engineer, the purpose would not of itself be illegal, nor necessarily undesirable. Prohibiting reverse engineering to discover programming ideas would do violence to the fundamental concepts of copyright. If it is ideas which those opposed to reverse engineering wish to protect, then computer software requires some other form of legal protection altogether, such as patents which do grant monopolies over ideas. However, for reasons which we discussed above, that would choke off the step-by-step development of the software industry.

The prohibition against reverse engineering is really being put forward as a substitute for the inquiry whether the software produced from the reverse engineering is a copy of the original software. As mentioned above, that will depend on whether the later software is “substantially similar” to the original software, which can be a very difficult and uncertain exercise: does one count the number of replicated codes, and if so, what proportion will result in substantial similarity; is it the overall “look and feel” of the two programs; should similarity of non-literal aspects be taken into account? Prohibiting reverse engineering outright avoids these difficulties by turning the existing law on its head: the act of reverse engineering is the primary copyright offence, and the close similarity of the two programs is nothing more than circumstantial evidence of the act of reverse engineering having taken place (Conley and Bryan, 1985, p. 597: Note, 1986, pp. 515-520). While the problems of proving substantial similarity are not to be minimised, the solution of totally prohibiting reverse engineering is worse than the difficulties it is designed to cure, for the policy reasons discussed above.

The EC Commission and the Economic and Social Committee did not specifically deal with reverse engineering in their drafts of the Directive, but the very broad definition of reproduction requiring authorisation of the copyright owner would prohibit the reverse engineering
programmer from even loading or running the software on his or her computer (CEC, 1989, art. 4). The EC Parliament was concerned that such a thorough-going ban on reverse engineering would entrench the dominant position of large, mainly non-European, software houses, by allowing them to lock off blocks of customers behind their indecipherable interfaces. The Parliament has recommended that the Directive provide as follows: 

The legitimate owner of a copy of a program may, without having to request the authorisation from the rightholder, observe, study or test the working program in order to determine its underlying ideas, principles and characteristics whether these are not protected by copyright, in the course of loading, viewing, running, transmission or storage in the execution of his contractual duties (CEC, 1989, art. 5(2a)).

However, the absence of a specific reference to decompiling, suggests that the reverse engineering party only has a "right to peer" into the software, and not a right to comprehensively disassemble it. As discussed below, this more limited right is to be contrasted with the more explicit right of users to reverse engineer software in order to achieve interoperability with their other software, provided that does not undermine the commercial rights of the copyright holder.

In our view, the argument over reverse engineering confuses the real issue: whether the disassembly leads to the creation of a final product which is not substantially similar to the original program. Copyright law cannot consistently with its basic principles prohibit reverse engineering, either directly or indirectly.

4. ADAPTATION TO IMPROVE/FACILITATE USE

Software, particularly more sophisticated and industrial application software, is often adapted by users to improve performance and efficiency of the program or to achieve a better fit with their existing hardware or operating environment.

In developing a rule to deal with adaptation, both the EC’s Green Paper and the CLRC’s Issues Paper refer to the need to balance “the needs of the user and the owner of the program... so that the user can extract the maximum use from the program while not adversely affecting the rights of the owner” (CLRC, 1990, p. 19). The EC Green Paper suggested that “adaptation which improved software’s efficiency when used within the scope of the licence provisions agreed between the user and the supplier should be considered as a legitimate and even necessary aspect of a user’s right to use the program for the purposes for which it was acquired... the supplier’s consent should not be needed or is rather to be conclusively presumed” (CEC, 1988, para. 5.6.15). On the other hand, adaptions which conflict with the supplier’s “normal exploitation of his rights should be prohibited, including translating the program to enable it to run on a machine other than that for which it is licensed, the commercial sale of adaptions” (para. 5.6.15).

The European Commission reversed tack in the draft Directive, prohibiting adaptation. The Commission explained that there were two imperatives for the owner maintaining strict control of adaptions (CEC, 1989, C91/12). First, warranties and maintenance arrangements would be invalidated or rendered expensive or impossible. Second, the owner will frequently set the licence rate to account for use of the software, and insert metering or anti-copying systems in the software. If the user could adapt the program, he or she would be at liberty to remove these mechanisms. However the European Commission’s real concern may lie with its prediction about the future direction of the software industry. While American analysts believe that the growth of package software is at the expense of custom software and processing services, European commentators believe that there will be a significant shift to integrated solutions for each user which involve supplying different combinations of application programs that are more easily adaptable and portable than in the past. Extensive user rights to adapt could undermine this new direction (CEC, 1988, para. 5.2.9).

The EC Parliament has returned to the thinking in the Green Paper and recommends that the Directive permit a licence holder or its contractor to copy or adapt a program where essential to its maintenance or the creation or operation of interoperability, subject to the following conditions:

(a) the information necessary to achieve interoperability shall not have been published or made available previously;
(b) the retrieval of information shall be confined to parts of the original program which are necessary for the achievement of the permitted aim;
(c) the information retrieved may not be communicated to third parties except insofar as this is necessary for the operation of the second program;
(d) the information retrieved may not be used to create or market a program which violates a copyright or the program of origin;
(e) this right will not be used in a way which allows the information obtained to be used in a way which unjustifiably damages the legitimate interests of the rightholder or which is contrary to the normal operation of the program (CEC, 1989, art. 5a).

The sale of packaged software imported from overseas constitutes a significant sector of the Australian industry. The importer/vendor may not have the skill or the interest in providing an extensive modification service. In our view, there is justification for a narrowly drawn adaptation right along the lines suggested by the EC to allow Australian users to modify the packaged or shrink wrap software to their needs.

5. CONCLUSION: A NEW WAY FORWARD?

As should emerge from the above discussion, the applica-
tion of the ideas/expression dichotomy to the hierarchy of programming elements is a difficult and inscrutable task. At the very least, the definition of software in the Australian Copyright Act should be amended to include a statement along the lines of the original draft of the EC Directive and Section 102(b) of the US Copyright Act that principles, logic, algorithms, format and structure and interfaces represent ideas and are not protectable. Further, we think that it would also be useful to expressly provide in the Copyright Act, much along the lines of section 11 of the Circuit Layouts Act that otherwise protectable programming elements will not be protected if they are commonplace in the industry.

However, clarifying provisions along these lines has not saved US courts or the EC from confusion and conflicting views over where the line is appropriately drawn. The better course may be to go back to first principles, and to look afresh at the "policy algorithm" which any law protecting computer software seeks to implement. This policy algorithm can be fairly uncontroversially stated as ensuring sufficient incentive and reward for the intellectual efforts devoted to the development of software, while at the same time leaving it open for subsequent programmers to build on developments in the writing of their own programs.

One recent US commentator has argued that these goals are still best achieved within the framework of copyright, but software is more sensibly analogised with works which are compilations of facts, such as directories, rather than with creative literary works:

Like an unadorned map or directory, a computer program is valued according to objective criteria, not aesthetic ones. The value of a map or directory, if it lacks originality, lies in its accuracy and thoroughness. The value of a computer program that communicates only with a machine lies in its speed and efficiency . . . The process of creating work that is to be appreciated for its aesthetic appeal is guided by something internal to the author, while the process of finding the best method of accomplishing a predefined task is guided by the external laws of mathematics, logic and applied science. The process of programming is thus far more like the process of fact gathering, which is constrained by the external world, than it is like the process of creating objectives with aesthetic value (Dayan, 1990, pp. 273-274).

The compilation cases have given more emphasis to the degree of labour, skill and investment put into a work than cases dealing with copyright in creative works (Lahore, 1989, p. 7183; Gorman, 1963a, 1963b). The plaintiff will have copyright in the compilation where he or she has devoted significant efforts in gathering the facts and arranging them in some order in the compiled work. Conversely, there is no copyright in facts, and a subsequent compiler of the same facts will not infringe copyright if he or she has also devoted sufficient efforts in coming to his or her own arrangement of those facts, even if he or she had resorted to the earlier compilation.

From the US compilation cases, the commentator extracts a four part test:

(a) The court should consider how 'industrious' the plaintiff's enterprise of gathering and compiling is; that is, how much labour and expense is invested in the production of the plaintiff's products. This factor is designed to ensure that the plaintiff has done enough work to merit setting the judicial machinery into operation (Dayan, 1990, p. 266). Most software would pass this first factor, other than those routines which are commonplace or fairly mechanistic.

(b) The court should also consider the speed, cost and extensiveness of the defendant's particular method of copying the results of the plaintiff's labour. The purpose of the inquiry is to determine whether failing to protect the plaintiff will make it so difficult for those in the plaintiff's position in the future to earn a fair rate of return that the product might not be produced (Dayan, 1990, p. 266). Literal transcription of large slabs of the object or source code would be viewed unfavourably under this factor, but reverse engineering, as properly understood, would not necessarily be viewed as suspect for the reasons discussed above.

(c) The court should also consider whether the defendant's method tends to add value to either the information copied or the fund of knowledge generally. Consideration should be given to the extent to which the defendant has used the information in combination with other information to advance knowledge in a given field (Dayan, 1990, pp. 269-270). If there has been a "wholesale appropriation" of the plaintiff's work and the defendant's work offers nothing new, that should count against it. The factor recognises the "stepping-stone" nature of advances in the computer industry, and would permit subsequent programmers to incorporate elements of an earlier program if the subsequent programmer can solve a problem the first did not or can solve it in a better way. Non-literal copying would also probably not fall foul of this factor, except in the circumstances of the Whelan v. Jaslow case where the subsequent programmer had translated the earlier program into another language without adding any original elements.

(d) The extent to which the defendant is competing against the plaintiff is a relevant factor because it is fierce competition, rather than small-scale "free-riding", that most threatens the profitability of the plaintiff. If the defendant is using elements of the plaintiff's work in a non-competing work, he or she should not be judged an infringer. However, competition will not be dispositive, though, when a competing defendant has added some value to the plaintiff's work (Dayan, 1990, p. 271). Pirating of software would clearly fail this test, but 'cloning' may not: while clones compete head-to-head with the original software,
clones are usually not a complete replication, but incorporate improved or additional features. As experience would appear to demonstrate, cloning, understood in this sense, is not so exploitive that it would drive software houses from the market, but rather enhances consumer choice.

Underlying this four part test is the view that some level of copying is a permissible mode of competitive conduct, as long as the resulting product reflects a sufficient level of effort and innovation by the copyst-competitor (Raskins, 1985, p. 338). A similar approach is found in the Australian Circuit Layouts Act, and its US progenitor (17 USC ss. 901-914). Section 23 of the Act provides that rights in a chip are not infringed by making a copy or copies of the layout for the purpose of evaluating or analysing the layout, and by making an original circuit layout based on an evaluation or analysis carried out with the use of these copies. Section 11 provides that a circuit layout will not be taken to be original if its making involved no creative effort or it was commonplace when it was made.

This suggested approach is not without its own problems, nor is it entirely consistent with the US compilation cases (Rockford v. Directory, 1985), let alone Anglo-Australian case law (Morris v. Wright, 1866, pp. 701-702; Ladbroke v. William Hill, 1964). Replacing the ideas/expression dichotomy with some relative weighing of the efforts put in by the original and successor programmers may be trading one vague abstraction for another. However, this approach at least has the virtue of focusing more clearly and tangibly on what is sought to be protected: the intellectual effort which is put into developing software.

Australia's ability to strike out in a new direction may be fairly limited because we are so heavily dependent on foreign software. As the EC debate on reverse engineering demonstrates, a significant shift from the known, even with its inherent difficulties, into the unknown is likely to be strenuously resisted. A significantly different and unfamiliar approach in Australia may threaten our attractiveness as a destination for foreign software. The outer boundaries for reform of Australian law may well be marked by the EC Directive, and even then the debate is likely to be ferocious.

REFERENCES
Articles, Books and Reports

UNITED STATES COPYRIGHT OFFICE (USCO): "Copyright Protection for Computer Programs", Copyright Office Circular R61, reprinted in 1 Copyright L Rep (CCH) 15,038.

Cases
Dyason v. Autodesk Inc., Federal Full Court (unreported 14 September 1990), Beaumont, Lockhart and Sheppard JJ.
Morris v. Wright (1866) LR1 Eq 697.

Statutes
Circuit Layouts Act (Cth) 1989.
Copyright Act (Cth) 1968.
Copyright Act (US) 17 USC (1988).

BIIOGRAPHICAL NOTES
Peter G Leonard is a partner with the Sydney law firm, Gillett & Tobin, Technology Lawyers, and specialises in computer law, telecommunications regulation and contracting, information technology and the media. He was previously a partner with Sly and Russell, founding that firm’s media communications division.

Peter Waters is a solicitor atGillett & Tobin where he specialises in computer law and contracting and telecommunications and regulation. As a Fulbright Scholar, he obtained his Master of Laws from Harvard University. He subsequently worked with a Washington law firm, specialising in international transactions and international taxation.
## CONTENTS

### Contributions

**1. February, 1990**

- **Suffern, K.G.**
  - An octree algorithm for displaying implicitly defined mathematical functions
  - Pages: 2-10

- **Caelli, W.G., Edmond, D., Knight, E.A. and Tickle, A.B.**
  - Implications of the tax file number legislation for computer professionals
  - Pages: 11-20

**2. May, 1990**

- **Goddard, J.**
  - Guest editorial
  - Pages: 25-26

- **Clarke, R.**
  - Social implications of IT - the professional's role
  - Pages: 27-29

- **Little, S.E. and Sauer, C.**
  - Social aspects of computing in computing education
  - Pages: 30-35

- **Morrison, P.R. and Forester, T.**
  - Teaching computer ethics and the social context of computing
  - Pages: 36-42

- **Coldwell, R.A.**
  - Some social parameters of computer crime
  - Pages: 43-46

- **Hughes, G.**
  - Computer crime; liability of hackers
  - Pages: 47-50

- **Graham, P.**
  - A case study of computers in public administration: The Australia card
  - Pages: 51-58

- **Whiteley, A.M. and Jordan, E.**
  - Behavioural aspects of introducing change in information systems
  - Pages: 59-68

**3. August, 1990**

- **Dengate, G., Couger, J.D. and Weber, R.**
  - Motivational characteristics of Australian information systems personnel
  - Pages: 77-87

- **Swatman, P.A., Swatman, P.M.C. and Everett, J.E.**
  - Stages of growth of an innovative software house: An additional criterion for software package selection
  - Pages: 88-98

- **Watson, R.T., Pitt, L.F. and Carey, N.**
  - Information systems technology in support of distribution: Some Australian experiences
  - Pages: 99-110

**4. November, 1990**

- **Clarke, R. and Greenleaf, G.**
  - Guest Editorial
  - Pages: 113

- **Wong, G., Kong, A. and Ngai, D.**
  - A study of unauthorized software copying among post-secondary students in Hong Kong
  - Pages: 114-122

- **Waters, P. and Leonard, P.**
  - The lessons of recent EC and US developments for protection of computer software under Australian law
  - Pages: 123-129, 131-132

- **Wiebe, A.**
  - Intellectual property protection of user interfaces in the USA and Germany
  - Pages: 133-139

- **Burnside, J.W.K.**
  - Copyright and computer software - Autodesk Inc. v. Dyason and Kelly
  - Pages: 140-144
### TITLE INDEX

(Titles in italics are for book reviews)

<table>
<thead>
<tr>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>A case study of computers in public administration: The Australia card</td>
<td>51-58</td>
</tr>
<tr>
<td>A Computational Logic Handbook</td>
<td>21</td>
</tr>
<tr>
<td>Ada for Distributed Systems</td>
<td>70</td>
</tr>
<tr>
<td>Advanced Turbo-C Programming</td>
<td>70</td>
</tr>
<tr>
<td>An octree algorithm for displaying implicitly defined mathematical functions</td>
<td>2-10</td>
</tr>
<tr>
<td>Artificial Intelligence and Expert Systems</td>
<td>24</td>
</tr>
<tr>
<td>Artificial Intelligence and Turbo C</td>
<td>148</td>
</tr>
<tr>
<td>A study of unauthorized software copying among post-secondary students in Hong Kong</td>
<td>114-122</td>
</tr>
<tr>
<td>Behavioural aspects of introducing change in information systems</td>
<td>59-68</td>
</tr>
<tr>
<td>Case Based Planning : Viewing Planning as a Memory Task</td>
<td>147</td>
</tr>
<tr>
<td>Challenging Mathematical Problems with (Basic, Pascal, Basic/Pascal)</td>
<td>23</td>
</tr>
<tr>
<td>Compiling Functional Languages</td>
<td>76</td>
</tr>
<tr>
<td>Computer Addiction? A Study of Computer Dependency</td>
<td>147</td>
</tr>
<tr>
<td>Computer crime; liability of hackers</td>
<td>47-50</td>
</tr>
<tr>
<td>Computer Integrated Testing</td>
<td>74</td>
</tr>
<tr>
<td>Computers and Social Change</td>
<td>72</td>
</tr>
<tr>
<td>Computers in the Human Context: Information Technology, Productivity and People</td>
<td>74</td>
</tr>
<tr>
<td>Computing Standards: A Practical Guide for Data Processing Management</td>
<td>147</td>
</tr>
<tr>
<td>Copyright and computer software - Autodesk Inc. v. Dyason and Kelly</td>
<td>140-144</td>
</tr>
<tr>
<td>Curves and Surfaces for Computer Aided Geometric Design</td>
<td>73</td>
</tr>
<tr>
<td>Digital Signal Processing: Applications to Communications and Algebraic Coding Theories</td>
<td>21</td>
</tr>
<tr>
<td>Discrete Optimization</td>
<td>70</td>
</tr>
<tr>
<td>Formal Description Techniques</td>
<td>69</td>
</tr>
<tr>
<td>Formal Methods in Artificial Intelligence, Cambridge Tracts in Theoretical Computer Science</td>
<td>69</td>
</tr>
<tr>
<td>Formal Methods: Theory and Practice</td>
<td>72</td>
</tr>
<tr>
<td>Fourth Generation Data: A Guide to Analysis for New and Old Systems</td>
<td>72</td>
</tr>
<tr>
<td>From Surfaces to Objects - Computer Vision and Three Dimensional Scene Analysis</td>
<td>75</td>
</tr>
<tr>
<td>How It Works ... The Computer</td>
<td>21</td>
</tr>
<tr>
<td>Hypertext - Theory into Practice</td>
<td>22</td>
</tr>
<tr>
<td>Implications of the tax file number</td>
<td>11-20</td>
</tr>
<tr>
<td>Information systems technology in support of distribution: Some Australian experiences</td>
<td>99-110</td>
</tr>
<tr>
<td>Intellectual property protection of user interfaces in the USA and Germany</td>
<td>133-139</td>
</tr>
<tr>
<td>Introduction to Computers</td>
<td>21</td>
</tr>
<tr>
<td>Knowledge Systems Design</td>
<td>24</td>
</tr>
<tr>
<td>Machine Vision for Inspection and Measurement - Perspectives in Computing (Volume 24)</td>
<td>73</td>
</tr>
<tr>
<td>Man-Computer Interfaces: An Introduction to Software Design and Implementation</td>
<td>112</td>
</tr>
<tr>
<td>Managing with Micros</td>
<td>111</td>
</tr>
<tr>
<td>Title</td>
<td>Pages</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Motivational characteristics of Australian information systems personnel</td>
<td>77-87</td>
</tr>
<tr>
<td>Pascal-SC A Computer Language for Scientific Computation</td>
<td>71</td>
</tr>
<tr>
<td>People and Computers III: Proceedings of the Third Conference of the British Computer Society</td>
<td>75</td>
</tr>
<tr>
<td>Principles of Optimal Design: modelling and computation</td>
<td>75</td>
</tr>
<tr>
<td>Social aspects of computing in computing education</td>
<td>30-35</td>
</tr>
<tr>
<td>Social implications of IT - the professional's role</td>
<td>27-29</td>
</tr>
<tr>
<td>Some social parameters of computer crime</td>
<td>43-46</td>
</tr>
<tr>
<td>Stages of growth of an innovative software house: An additional criterion for software package selection</td>
<td>88-98</td>
</tr>
<tr>
<td>Teaching computer ethics and the social context of computing</td>
<td>36-42</td>
</tr>
<tr>
<td>The Inverse Gaussian Distribution: Theory, Methodology and Applications</td>
<td>23</td>
</tr>
<tr>
<td>The lessons of recent EC and US developments for protection of computer software under Australian law</td>
<td>123-129, 131-132</td>
</tr>
<tr>
<td>The Mathematical Structure of Raster Graphics</td>
<td>147</td>
</tr>
<tr>
<td>The MSDOS Encyclopedia</td>
<td>71</td>
</tr>
<tr>
<td>Visual Editing on UNIX</td>
<td>22</td>
</tr>
<tr>
<td>XSEL's Progress: the Continuing Journey of an Expert System</td>
<td>75</td>
</tr>
<tr>
<td>Contributor</td>
<td>Pages</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>Caelli, W.J.</td>
<td>11-20</td>
</tr>
<tr>
<td>Clarke, R.</td>
<td>27-29,113</td>
</tr>
<tr>
<td>Coldwell, R.A.</td>
<td>43-46</td>
</tr>
<tr>
<td>Crossley, J.N.</td>
<td>21</td>
</tr>
<tr>
<td>Couger, J.D.</td>
<td>77-87</td>
</tr>
<tr>
<td>Dengate, G.</td>
<td>77-87</td>
</tr>
<tr>
<td>Edens, P.</td>
<td>147</td>
</tr>
<tr>
<td>Edmond, D.</td>
<td>11-20</td>
</tr>
</tbody>
</table>
Emerging as one of the most important issues in intellectual property law is the protection of user interfaces. The leading economic position of the U.S. in the software market is reflected by its forming the issues in the legal classification of this new technology.

A comparative study reveals the impact of the American legal development on German law as being that many emerging principles of law in this field can be availed of in a civil law country as well. In copyright law, this concerns the dichotomy of idea and expression and the classification of various elements of the user interface accordingly, regarding the user interface as a work separate from the program code. The same is true for the identification of unprotectable functional elements of the user interface. Copyright protection may be supplemented by other intellectual property schemes, like patent law.

The scope of protection provided for user interfaces accordingly will have a major impact on the economic development in this software field, including considerations of standardizations and compatibility. Policy consideration suggest a rather limited scope of protection.

Keywords: intellectual property, software copyright, patent, user interface, look and feel.
CR Categories: K1, K4, K5.1.
sets of code. Nothing is revealed by the legislative history as to the protection of screen displays and the definition of computer programs in § 101 of the Copyright Act is confined to the code (17 USC § 101 (1982)).

2.2 Case Law

2.2.1 Video Games

The protectability of screen displays has first become an issue in connection with video games. The protection of the underlying program has been held not to encompass the screen displays and thus to provide no protection against “knock-offs” with the screen displays constituting separate audiovisual works, § 102(a)(7) of the Copyright Act (Stern v. Kaufman, 1982). Three major objections against protection as a separate work were discussed by the court in Stern: the audiovisual displays are too transitory to be regarded as “fixed”, the creative participation of the player renders every game a separate original work and the interaction between player and program prevents the displays from being regarded as a “series of related images” as provided by § 101 (p. 855-857). The court responded by emphasising that the displays and sounds were repeated often enough to refute all three arguments.

2.2.2 Nongame Screen Displays

Pinheiro and Lacroix (1987, p. 412) refer to experts for the prediction that producers will regard protecting user interfaces as more important in the future than protecting code. An early case touching the issue in Synercom v. University Computing (1978) concerned the copyrightability of input formats which perform a function in inputting data comparable to screen display formats. Plaintiff had developed a program for the structural analysis of buildings that required the input of data in a certain sequence. The user handbook explained the system for arranging data and contained blank forms resembling the punched cards actually used. The formats by their placement of lines, shaded areas, and words determined the sequence of entering data. Defendant developed a computer program requiring the input of data in exactly the same sequence as plaintiff’s program, accomplished by way of a preprocessor program which was essentially a transfer of the blank forms to a computer program. The court found no infringement with the defendant having appropriated only the idea expressed by the formats which was the sequencing and ordering of data. Copyrightability of the blank forms was also denied as the idea of the sequencing of data and the expression had merged, above and beyond which the court found no stylistic creativity involved (Synercom, p. 1013-1014).

In the landmark decision of Whelan however, copyright protection was extended to structure, organisation and sequence of a program with important effects on the protection of user interfaces as well. The court rejected Synercom and referred to the definitions of “compilation” and “derivative work” as demonstrating the legislative intent to regard “sequencing and ordering of materials” as protected expression (Whelan v Jaslow, 1986, p. 1239). It emphasised the efforts in creating the program and the importance of providing for incentives by protecting the most valuable contributions of the author. While regarding program and audiovisual display as constituting two separate works, the court based the finding of infringement of the program structure on the similarity of the screen displays as indirect evidence (p. 1237).

This resulted in some confusion especially in the following two leading decisions on the protection of nongame screen displays. In Broderbund, plaintiff was the copyright owner of a menu driven program, originally designed for use on Apple computers, that permitted its users to create customised greeting cards, signs, banners and posters which allegedly had been infringed by an IBM-version of defendant (Broderbund Software v. Unison World, 1986). Separating idea from expression, the court focused on the function underlying the program which was the creation of greeting cards and found no merger of idea and expression (Broderbund, p. 1132). The court found infringement on the basis of similarities in the sequencing of the screens and their layout, the choices presented to the user, in short the “total concept and feel” of the program.

The court relied on Whelan for the proposition that copyright protection for the program extended to the screen displays. Even with the Copyright Office having issued a separate registration for the screen display as an audiovisual work, the court regarded program and output not as separate works but extended the “look and feel” of the program to the visual displays (p. 1132, 1137).

The purely textual status-screen of a command-driven data communication program was at issue in the second decision on user interfaces, a decision which seems to reflect a more appropriate analysis (Digital v. Softklone, 1987). The upper half of the screen reflected the present status of the operation parameter while the user could change the “settings” by inputting one of the commands listed in a window in the lower half. The status screen was registered as a “compilation of program terms”. Defendant produced a similar program with a mostly identical screen and a largely identical command language.

The court regarded the user interface as a literary work in that it was a compilation of the commands. It rejected the broad reading of Broderbund and expressly distinguished between the copyright of the program and the screen displays, ultimately relying on the fact that an identical screen could be created by separate and independent programs and thus could not be regarded as a “copy” of the program (Digital v. Softklone, p. 456).

In a detailed analysis of the protected elements of the screen display, the court focused on the process or manner in which the screen operated and not on the function of the underlying program, as Broderbund had done. It identified
not one single idea but a number of underlying ideas such as the use of the screen to reflect the status of the program, the use of a command driven program, and the typing of two symbols to activate a specific command (Digital v. Softklone, p. 459). With the command set being unprotected, the manner in which the command terms were presented, i.e. the arrangement and highlighting of two characters, was regarded as protected expression, capable of being expressed in a variety of ways. Defendant’s argument based on the importance of standardisation in the computer industry was rejected.

The increasing number of lawsuits in this field reflects the growing importance of this issue in the software market. They will contribute to a clarification of the state of copyright protection for the user interface.

2.3 Classification and Scope of Protection

2.3.1 Categories of Works

The case law discussed in this section provides a basis for a classification of user interfaces in copyright categories for the purpose of an overview of the current US position.

As has been demonstrated, the case law is divided as to whether user interfaces constitute just one form of expression of a program besides others (Johnson Controls v. Phoenix Control Systems, 1989) or whether program and user interface constitute two separate copyrightable works (Manufacturers Technologies v. Cams, 1989). As Reback and Hayes (1987, p. 8) point out the source code as well as the data for creating the same screen display may vary considerably depending on different factors, e.g. different display techniques. In my view this is the main reason for treating user interfaces as separate works. Program and screen display not only give the impression of different authorship but also are often created by separate authors as concerns training and skills, and their design requires substantial investment. Who is supposed to be the owner of the copyright on the user interface in case of different authors for the audiovisual work and the code? Both have different functions. Application programs are intended to implement the program functions, user interfaces make a dialogue with the user possible, each perhaps in different languages. Therefore, user interfaces, like video game displays, ought to be regarded as separate works for copyright purposes.

2.3.1 (a) Audiovisual Works

Screen displays could be classified as audiovisual works. If the screen consists only of a few words or phrases a lack of sufficient originality is probable but may be overcome if graphic elements are included which come closer to the complexity and variety of video games. If the screens qualify as audiovisual works, textual elements are included within the scope of this protection (WGN Continental Broadcasting v. United Video, 1982).

2.3.1 (b) Pictorial or Graphic Works

Screen displays may also be regarded as “pictorial or graphic works” (17 USA § 101). Thus, as Pinheiro and Lacroix (1987, p. 434) show, windows, icons and menu bars may be regarded as “graphics”.

2.3.1 (c) Literary Works

A user interface which is purely or mostly based on textual elements may also be classified as a literary work in the category of “compilations” (Digital v. Softklone, 1987, p. 463). As far as the selection is made by the author and does not appear to be completely arbitrary or dictated by restraints, the requirement of originality is satisfied. This may not be true for simple words, phrases, or symbols or short lists of commonly used names, e.g. the command “type” to which Pinheiro and Lacroix (1987, p. 433) refer.

A command language as such may receive protection as a literary work. The Softklone court regarded the use of an identical command language without a particular presentation on the screen as an idea and found no infringement (Digital v. Softklone, p. 460). Middleton (1988, p. 219) separates the idea of using a command language from the unique compilation of commands which may be quite creative and manifold and ought to be regarded as protected expression. The command set may also often be encompassed by the copyright on the user interface or the program code.

The scope of protection is to be refined by the application of the idea/expression-dichotomy and the useful article doctrine.

2.3.2 The Idea/Expression-Dichotomy

The copyright balance struck between the conflicting interests and the definition of the scope of protection is implemented by the principle that protection extends only to the expression and not the idea of a work (Baker v. Selden, 1879). The means to separate idea from expression is the test whether the idea can be expressed in a variety of ways or in only one or a few ways in which case the expression is said to merge with the idea (Apple v. Franklin, 1983).

In Whelan the court identified the purpose or function of a utilitarian work with the work’s idea and the existence of various means of achieving the desired purpose rendered the particular means chosen not necessary to the purpose and hence protectable expression (Whelan v. Jaslow, 1986, p. 1236). The court defined the idea of the program as the efficient management of a dental laboratory and the expression as “the manner in which the program operates, controls and regulates the computer in receiving, assembling, calculating, retaining, correlating and producing useful information either on a screen, print-out or by audio-communication” (p. 1239), rendering structure, sequence and organisation protectable expression.

While this reflects current law, this expansion of protec-
tion has been heavily criticised, eg. by Sutton (1987, p. 563) both as extending protection too far to the underlying function of a program and as to the danger of manipulating the scope of protection resulting from the definition of the program's function.

Applying Whelan to user interfaces, it seems appropriate to focus on the user interface and its function as a means of communicating with the program. The real problem is defining the idea and separating idea from expression in a specific case. Elements related to how the program receives commands and reflects the results and how the computer operates should be regarded as ideas. Thus, the provision of menu choices may be regarded as idea and more ideas may be found pursuant to limited methods of interaction on the next lower level as did the court in Sofiklone for the use of the first two letters to activate a command (Digital v. Sofiklone, p. 459). The protectability of other elements also depends on the possible range of expression.

The exact line of demarcation remains unclear, however. Thus, the use of trash can icons may be regarded as one of several possibilities of expressing the "delete"-function as Pinheiro and Lacroix (1987, 429) do. As Kullby (1988, p. 882) discusses, it may also be regarded as an idea, however, leaving the possible number of expressions very limited.

Other considerations have to be included. There is only a limited number of input methods and users get accustomed to specific commands. The requirement of inputting the same data in a logical way may necessitate the selection of similar input data and sequence of screens and limit the possible range of variations, perhaps to the field of decorative graphics (Note, 1987, p. 126).

Moreover, "human factor analysis" has established itself as a scientific discipline analysing the user/computer-communication and Menell mentions several design objectives that are widely being pursued (Menell, 1989, p. 1054) and may limit the variety of expressions. All this may mitigate in favour of very limited copyright protection for user interfaces.

2.3.3 The Useful Article Doctrine

Another fundamental principle of copyright law is that protection may exist in the description of useful articles or processes but not in the articles or processes themselves, which may constitute patent subject matter (17 USC § 113 (b)). A useful article is protected only to the extent that the artistic features can be identified separately and are capable of existing independently of the functional elements (Mazer v. Stein, 1954) even if the appearance of an article is dominated by aesthetic features (Fabrica v. El Dorado, 1983).

The function of a user interface is to facilitate the input and output of data. As Pinheiro and Lacroix (1987, p. 435) point out, it appears to be too simplistic to isolate the useful functions within the hardware with the aesthetic elements being presented on the screen. User interfaces are not only a means of communicating to the user which information the system requires, but also provide a means of enabling the user to input those data. Protection then depends on the extent to which the artistic elements are separable from the useful elements, which will be more likely with graphic oriented than with text oriented screens. This doctrine also suggests a rather limited scope of protection.

3. PATENT PROTECTION IN THE USA

The presence of useful elements in a user interface raises the question of the availability of patent protection. American patent law provides for more far-reaching protection than German law, which denies patents mainly on the ground that software is outside the realm of the technological arts. In the US, the discussion focuses on the exclusion of mathematical algorithms (In re Pardo, 1982). As Ruddy-sill et al (1989, p. 15-16) report, patents have been obtained for database management systems as well as expert systems and recently even in the field of user interfaces.

For example, one patent obtained by IBM relates to the invention of an improved data input screen of an interactive data input system (US Patent No. 4 646 250 — "Data Entry Screen"). The first screen displays the field for inputting data in a "highlighted" manner while optional fields are displayed in a normal way. After data input, incorrect information, e.g. with wrong ZIP code, is displayed again in a "highlighted" manner.

Further patents include a "touch sensitive" data entry screen (US Patent No. 4 672 558) a "touchscreen" for the two-dimensional emulation of three-dimensional objects (US Patent No. 4 649 499) and a windowing technique (US Patent No. 4 586 035). As Menell (1989, p. 1092) reports, several design patents have been issued recently covering various design elements of user interfaces.

4. COPYRIGHT PROTECTION IN GERMANY

The above overview of the state of American law has demonstrated that the protection of user interfaces is already a heavily discussed issue in the US. The same is not true for German intellectual property law, but due to the increasing importance of user interfaces in the marketplace this seems to be only a question of time.

Thus, as in other advanced countries, the need arises to integrate this subject matter into the intellectual property scheme. While Germany is a civil law country, intellectual property law is subject to statutory regulation in both countries and to international treaties which are a force for worldwide standardisation of the law. This reduces problems of comparative law and enables German law to profit from the advanced American discussion.

The copyright classification of user interfaces in Germany is not supported by existing video game cases.
games are protected as motion picture works pursuant to § 2 Sec. 1 No. 6 or § 95 of the German Copyright Statute (UrhG) that requires a “sequence of pictures and sound giving the impression of a moving picture” which cannot be said even of a sequence of screen displays.

4.1 Literary Works or Scientific and Technical Representations
To qualify for protection as a literary, scientific or artistic work, a minimum requirement of aesthetic quality exists. This is met, as user interfaces also convey information to the user and there is no requirement of distinguishing functional elements in German law. Thus, screen displays may be classified as literary works pursuant to § 2 sec. 1 No. 2 UrhG or as a scientific or technical representation pursuant to § 2 sec. 1 No. 7 UrhG, depending on the prevalence of textual or graphic elements.

Comparable to the idea/expression-dichotomy in American law, it is a basic principle of German copyright law not to protect the contents of a scientific or technical work but only the expression which is located in the selection, collection, classification and arrangement of the information and subject matter. Also excluded from protection are elements that are common or indispensable for scientific reasons (Judgement of Bundesgerichtshof, W. Ger., 1981).

Individuality as the major requirement of German copyright law requires a certain amount of creativity. As a prerequisite, there must be room for creative authorship, which essentially means that there is a variety of possible solutions giving the specific solution an individual character which may be lacking for elements dictated by function. In a key decision in the software field, the Federal High Court has tightened the requirement to include a considerable rise of the intellectual and creative effort above the skills of the “average programmer” (Judgement of 9 May 1985, Bundesgerichtshof, W. Ger., 1985 — “Inkassoprogramm”, reported in English in 17 IIC 681 (1986)). This would leave a majority of programs unprotected and has resulted in copyright law being widely unavailing for the protection of computer software. There is widespread expectation, however, that this will not be the last word on the issue by the courts or the legislature.

The necessary delineation between content and expression suggests that the results of the idea/expression-dichotomy in American law could be useful in the process of defining the elements of user interfaces protectable under German copyright law, subject to the additional (if uncertain) requirement of creativity. Regarding the concepts and methods of selecting and communicating information as contents or ideas of user interfaces, the specific selection and combination of these methods in their form of representation may constitute protected expression. The margin of various modes of expression may be limited by the function of user interfaces and other factors referred to in the American idea/expression/discussion. In addition, widespread use of a specific interface may render certain elements, eg. a command language, common and lacking the necessary creativity. As discussed for the American law, these factors may limit the protection available for user interfaces.

4.2 Photographic Works
There are also other categories of German copyright law that may be taken into consideration. One would be protection as a photograph pursuant to § 2 sec. 1 No. 5 or § 72 UrhG which also includes TV pictures. Schulze has argued that regardless of method of creation even a programmed picture as a product similar to a photograph may qualify for protection in this category (1988, p. 190).

4.3 Artistic Works
Graphically oriented displays may qualify for protection as artistic works pursuant to § 2 sec. 1 No. 4 UrhG. This may include aesthetically appealing menus or single elements that fulfill the heightened requirement of individuality.

5. OTHER INTELLECTUAL PROPERTY SCHEMES
5.1 Industrial Design Protection
Beyond copyright, there are other intellectual property schemes available in German law. Below the level of creativity required in copyright law, protection may be obtained under the Industrial Design Protection Act (“Geschmacksmustergesetz”). This type of protection focuses on the aesthetic appeal of an industrial design or model. The model has to be registered with the Patent Office and protection is granted for five years with a possible extension up to 20 years. Technical or design elements dictated by function are not in themselves protectable but require an additional aesthetic content.

The required level of originality will be ascertained from the overall aesthetic impression of the model to which functional elements may contribute (Judgement of Bundesgerichtshof, W. Ger., 1966). The required level of individuality exceeds the ability of an average creator but to a lesser extent than required in copyright law. Copyright and design protection may subsist in the same work. There is an additional requirement of novelty in the sense of not being known to domestic experts.

As far as the copyright discussion revealed the existence of not necessarily functionally dictated elements, user interfaces may be regarded as aesthetically appealing in their overall impression. The other requirements, however, may seriously limit the protection available for industrial models or designs.

5.2 Typeface Protection
The Typeface Protection Law (1981 BGBl, part II, at 382) provides protection for novel and individual typefaces
including characters, numbers, and other tokens as well as ornamental elements redesigned to create texts through various graphic techniques. Registration is required and protection is granted for a period of 10 years with a possible extension up to 25 years.

Again, elements dictated solely by function are excluded from protection, as Kelbel (1982, p. 79) points out. Protection extends to the use of the protected typefaces for the production of texts by way of graphic techniques including the use of digital memory and processing. This form of protection applies to textual and ornamental elements of user interfaces as far as they can be regarded as being employed in the production of texts.

5.3 Patent Protection

The function of data input and output performed by user interfaces is regarded by German patent law not to be within the technological arts. Facilitating the use of a program, and conveying information about the program, they are regarded as instructions to the human mind which are excluded from patent protection as not being within the technological field. With the Federal High Court still pursing a point of novelty approach and regarding the running of a program on a computer system as essentially untechnical, there is little prospect for obtaining patent protection for user interfaces. The exceptions for programs integrated in the control of a technical environment or changing the internals of the hardware (Judgement of Bundesgerichtshof, W. Ger., 1977) are not applicable to user interfaces.

5.4 Utility Model Protection

The Utility Model Act provides protection with lower requirements than patent law. Basically, screen displays constitute subject matter eligible for this type of protection. As in patent law, however, there is a requirement of being within the technological arts that excludes protection in the same manner.

6. CONCLUSION

Although there are no court rulings available in German law yet, the attempt to classify user interfaces for intellectual property protection has demonstrated that there are more intellectual property schemes available in German law, while at the same time the requirements for obtaining protection are more demanding, resulting in a lesser degree of protection for user interfaces in German law compared to American law. This is true for the copyright as well as the patent type of protection. Regardless of these differences, it has become apparent that there are certain principles that are generally important for intellectual property protection in both countries. Here, German law is able to profit from the advanced American discussion on this subject.

In copyright law, as the main type of protection, this concerns the delineation between the idea or contents and the expression as well as the exclusion of functional elements that limit the possible range of expression. Based on an understanding of a user interface as a work separate from the program, the copyright principles discussed in American law considerably limit the protection available for user interfaces. Although the reported cases are contradictory to some extent, the subsequent analysis has provided guidelines and arguments for identifying specific protected and unprotected elements of user interfaces in the application of the law. If this analysis has turned out to be useful for a civil law country like Germany, as has been shown, it ought to be taken into consideration by other advanced countries as well.

Moreover, user interfaces constitute another product of new technologies that raises the question of whether and how much protection will promote or impede progress and innovation. This may include consideration of factors like standardisation and compatibility. User interfaces assume an increasing importance in marketing software. Widespread acceptance and training costs create the danger of a “lock-in”, as Karjala points out (1987, p. 70). In addition, innovation in the field of intellectual property is based on building upon the efforts of others. Extended protection may multiply development costs and erect additional barriers to market entry by the need to circumvent protected elements.

These policy considerations confirm the rather limited scope of protection suggested by the above analysis. The principles discussed may provide means of striking an appropriate balance between the competing interests of creators, competitors and the public.

REFERENCES

Publications


KELBEL, Der Schutz typographischer Schriftzeichen, 1982 GRUR 79.


PROTECTION FOR USER INTERFACES

Cases
Fabrica Inc. v. El Dorado Corp., 697 F.2d 890 (9th Cir. 1983).
In re Pardo, 684 F.2d 912 (CCPA 1982).
Judgement of Bundesgerichtshof, W.Ger., 1966 GRUR 97 ("Zundaufsatz").
Judgement of Bundesgerichtshof, W.Ger., 1977 GRUR 352 ("Staatsexamensarbeit").
WGN Continental Broadcasting Co. v. United Video Inc., 693 F.2d 622 (7th Cir. 1982).

BIOGRAPHICAL NOTE
Mr Wiebe is a staff member of Prof. Wolfgang Killian, Director of the Institut fur Rechtsinformatik (Computer & Law) at the University of Hanover. He was awarded the degree of Masters of law by the University of Virginia in 1988, and is currently completing his doctoral dissertation.
Copyright and computer software — Autodesk Inc. v. Dyason and Kelly

J. W. K. Burnside, Q.C.
Owen Dixon Chambers, 205 William Street, Melbourne Victoria 3000

In 1984, the Commonwealth Parliament amended the Copyright Act by introducing a specific reference to computer programs. Autodesk Inc. v. Dyason and Kelly is the first case in Australia to test the application of the Copyright Act to computer software, since the Act was amended.

The case raised fundamental questions about the nature of "computer programs" as defined in the Copyright Act, and the nature of the protection given to computer programs.

Historically, copyright protection has been granted to the form in which an idea is expressed, rather than to the idea itself. The attempt to protect computer programs by including them within the category of "literary works" for the purposes of the Copyright Act, challenges fundamental notions about the application of copyright law, and raises for consideration whether copyright is an appropriate medium for the protection of computer software in any event.

Keywords: intellectual property, software copyright, dongle.

CR Categories: K.1, K.4, K.5.1.

INTRODUCTION

In 1984, the Commonwealth Parliament amended the Copyright Act by introducing a specific reference to computer programs. Until that time, the Copyright Act was silent about computer programs. There had for several years been a lively debate about whether or not the Copyright Act extended protection to computer programs. In 1983, in the case of Apple Corporation v. Computer Edge Pty Ltd, the Federal Court of Australia decided that the Copyright Act did not protect computer programs. Beaumont J's decision went on appeal to the Full Federal Court. By a 2:1 majority, the Full Court held that the Copyright Act did provide some protection for computer programs. Whilst the matter was under appeal to the High Court, Parliament amended the Copyright Act.

Autodesk Inc. v. Dyason and Kelly is the first case in Australia to test the application of the Copyright Act to computer software, since the Act was amended. This paper reports on the original judgment and the results of the appeal, in this very important case.

THE FACTS

AutoCAD is a computer-aided drafting program created and marketed by Autodesk Inc. In Australia, it is supplied to customers on nine floppy disks, together with a hardware lock. The issues raised by the case focussed on the use of the hardware lock.

The purpose of the lock is to prevent AutoCAD from being run unless the lock is connected to the serial port of the computer. As only one lock is supplied when AutoCAD is purchased, the practical consequence is that only one copy of AutoCAD can be used by the customer at any one time.

A hardware lock is a small device which will sit in the serial or parallel port of a computer and will accept a corresponding lead to a peripheral at the other end. The lock is transparent to the peripheral device, but it can be interrogated from time to time by the computer. The purpose of the interrogation is solely to establish that the lock is present. If the lock is not present, the operation of the program will cease.

AutoCAD is marketed in a form which does not prevent its being copied as many times as the user wants — in fact, the manufacturer encourages such copying. The only practical limitation on the use of AutoCAD was the hardware lock. The Respondent Kelly, out of curiosity, decided to see what the AutoCAD lock did. Using a dual channel oscilloscope, he detected a pattern of bits passing from the lock to the computer in response to voltage transitions from the computer. He noted that, after a while, the sequence of bits passing from the lock to the
computer began to repeat. Ultimately, he detected that the response of the lock was a repeating 127 bit string. These observations involved no examination or analysis of the internal workings of the lock. Indeed, the evidence was that Kelly only learned how the AutoCAD lock worked when AutoCAD’s experts gave evidence at the trial.

Kelly determined to see whether he could make a device which would produce the same output as the AutoCAD lock. He decided that the simplest was to store the entire 127 bit sequence in an EPROM, with a counter to detect incoming transitions and advance to the next EPROM address. It worked.

Kelly then decided to see if there was a market for the device. He obtained advice from a patent attorney that his device involved no infringement of Autodesk’s legal rights. He enlisted the help of the Dyasons to market the device.

Autodesk was upset when it saw Kelly’s lock being advertised as able to run all versions of AutoCAD. It obtained Court orders in Kelly’s absence. The matter ultimately came on for trial before Northrop J. in the Federal Court at Melbourne.

The central issues at trial were:
(a) whether the AutoCAD lock was protected by copyright, and
(b) whether Kelly’s lock (“Autokey”) infringed that copyright.
A further issue was:
(c) whether running a program is an act comprised within the copyright.

COPYRIGHT
Copyright was first granted by statute in the early 18th Century. Its original purpose was to secure the economic interest of printers. It protected only the precise form of words used in a work, and prevented only a copying of those precise words. Later developments saw the protection extended to obvious imitations of the work (i.e. not just strict copies but “modified” versions) and also to protect other forms of work.

Since the emergence of personal computers, and the explosion in software marketing which followed, there has been increasing debate in various jurisdictions about whether copyright was an appropriate form of protection for computer software. The Apple Corporation was conspicuously active in many jurisdictions in testing exactly that question. The litigation in Australia of Apple Corporation v. Computer Edge Pty Ltd showed clearly that the existing copyright law provided at best only patchy and unsatisfactory protection.

The Copyright Act now protects various forms of work, including literary works, artistic works, sound recordings, cinematograph films and so on. The owner of the copyright has the exclusive right to do (or to authorise the doing of) any of the various acts comprised in the copyright. The copyright in literary, dramatic and musical works includes the right to reproduce the work in a material form, to publish the work, and to make an adaption of the work.

In 1984, the Federal Parliament amended the Copyright Act so as to make specific reference to computer programs. It did this primarily by altering the definition of “literary work” so as to include computer programs within the meaning of that expression. Anything which is a “literary work” attracts the copyright protection given by section 36. The Parliament defined “computer program” as meaning:

“an expression, in any language, code or notation, of a set of instructions (whether with or without related information) intended, either directly or after either or both of the following:
(a) conversion to another language, code or notation;
(b) reproduction in a different material form to cause a device having digital information processing capabilities to perform a particular function”.

“Material form” was defined as meaning:

“any form (whether visible or not) of storage from which the work . . . can be reproduced”.

It should be observed that a computer program is “the expression of a set of instructions . . . in any language code or notation . . . ”. One of the difficulties created by the case arose from the fact that both locks were “pure hardware” devices. They were not firmware. Clearly, both devices fall outside the common understanding of the expression “computer program”.

It is notable that the extension of copyright protection to new forms of work has generally been done with some care and consideration. A mode of protection appropriate to the subject matter has generally been devised and implemented. By contrast, the amendments to the Copyright Act in the wake of the Apple case represented a “quick fix” solution. Although it was better than nothing, it could not be regarded as a perfect, or permanent, solution. Even at the time, some of its deficiencies were apparent. For example, since the Copyright Act deems computer software to be a literary work, it provides protection for the life of the author plus 50 years. Such a long period of protection is demonstrably unnecessary for computer programs. The Act prevents the program from being converted into a dramatic form a restriction (it might be thought) which is unlikely to trouble even the software liberation movement. However it does not, in terms, say whether running a program is
an act comprised in the copyright, yet this is a serious (and undecided) question.

THE DEVICES
The AutoCAD lock contains a shift register and an exclusive-or gate. They are wired together so that pins 6 and 7 of the shift register provide the input to the exclusive-or gate. The output of the exclusive-or gate goes to pin 1 of the shift register. The content of pin 6 of the shift register is presented as output from the lock. A voltage transition from the computer stimulates the shift register, the contents of the shift register change, step by step, as incoming transitions are received. Position 1 of the shift register is replenished by the output of the exclusive-or gate; the changing contents of pin 6 are sent back to the computer.

The hardware arrangement in the AutoCAD lock is one commonly used as a pseudo random number generator. The AutoCAD lock contained no microprocessor and no memory chips. See Exhibit 1.

The Autokey lock, on the other hand, contains a counter and an EPROM. The least significant bit of each of the first 127 memory addresses contains each in turn of the 127 bits comprising the repeating pattern Kelly had observed. Each incoming voltage transition from the computer causes the computer to advance the EPROM to its next address. The least significant bit at that address is available as output to the computer. It contains no microprocessor. See Exhibit 2.

Thus, the two locks produce the same string of bits in different ways. The AutoCAD program contains a sub-routine, referred to as Widget.C. (Part of the source code version of Widget.C was put in evidence. Although the program is supplied in compiled form, the relevant portion of code was referred to throughout the hearing as Widget.C and this paper does the same.) It was the function of Widget.C to check the response of the lock to see if it was what was "expected". The evidence was that Kelly never looked at the AutoCAD program to see how it dealt with the lock's responses. In any event, the prospect of finding and interpreting that portion of compiled code which embodied Widget.C is vanishingly small.

Autodesk led evidence of the way in which Widget.C checked the lock's responses. Expert witnesses called on behalf of Autodesk said that the locks and Widget.C were each implementations of the same state machine — that is, each produced the same 127 bits. No evidence was led of any "set of instructions" on which the AutoCAD lock had been based.

This led to one of the great practical difficulties in the case. The essence of infringement of copyright is that the infringing work must objectively resemble the protected work.

It is of the essence of copyright that it does not protect ideas, but only the way in which those ideas are expressed. This has been said many times in many courts. As long ago as the 1st Century AD, the Roman philosopher Seneca said "Ideas are common property". More recently, English and American courts since the 18th Century have insisted that copyright protects only the form of expression and not the underlying ideas embodied in the work. It is for this reason that objective similarity is the test of infringement: if the same idea is expressed in a different way, it is not an infringement of another expression of that idea.

A substantial difficulty emerged in the law of copyright, however, when it was accepted that copyright in a work can be infringed not just by direct literal copying of the precise words used, but by less precise copying or adaptation. If copyright can be infringed by something less than wholesale copying of the precise mode of expression, how does the Court determine where the idea ends and expression begins?

This problem in copyright law is commonly referred to as the idea/expression dichotomy. The difficulty of applying the idea/expression dichotomy is well illustrated by considering the plot of a play. At
what level of description does the plot move from being a mere idea, the common property of mankind, to being a particular expression of that idea, thus the property of the author? The problem was stated as follows by Judge Learned Hand in 1930:

"... when the plagiarist does not take out a block [of text] in situ, but an abstract of the whole, the decision is more troublesome. Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his 'ideas', to which, apart from their expression, his property never extended .... Nobody has ever been able to fix that boundary, and nobody ever can ...."

Despite the difficulty of the exercise, Courts have held works to infringe other works if details of plot, incident and characterisation are substantially similar, although no words are copied.

American courts have already grappled with the idea/expression dichotomy in the context of computer programs. The so-called “look and feel” cases apply to software the notion of generalised (rather than specific) similarity which is familiar in connection with novels and plays. Instead of looking for similarity of plot, incident and characterisation, they look for similarity of aspects of the screen appearance and user interface — the look and feel of the programs. These can be similar without there being a corresponding similarity of the code which comprises the programs.

THE FINDING

After a hearing which lasted two weeks, Mr Justice Northrop held:

(a) that the AutoCAD lock was a computer program; and
(b) that the copyright in the AutoCAD lock had been infringed by the Autokey lock.

The first of these findings is surprising, but the second is startling in view of the fact that the hardware arrangements were quite different in each device. In a crucial passage in the Judgement, Mr Justice Northrop said:

"Counsel for the Respondents denied that ... there was a 'sufficient degree of objective similarity' between the Autokey lock and the AutoCAD lock so as to satisfy the requirements of ... the Copyright Act.

There is a difference in form between the hardware within the AutoCAD lock and the Autokey lock, namely the shift register and the EPROM. But the Court has concluded that each lock constitutes a computer program. Each performs the same function. It is this function which is the essential aspect of each lock. Function has a particular importance in the definition of a computer program and regard must be given to this concept of function in considering the question of whether there is a ‘sufficient degree of objective similarity’ between the two locks.”

This finding seems to involve a fundamental shift in the emphasis of copyright law towards the protection of ideas. The idea of each lock was to produce the same output of 127 bits. Each lock did so by quite different hardware arrangements, embodying different algorithms.

If the Judgement is right, one might ask rhetorically:

(a) would the Autokey lock still have infringed copyright in the AutoCAD lock, if the EPROM held a different series of digits: its hardware and logical processes would be unchanged, its output would be different;
(b) is it possible to build any hardware device which produces the right 127 bit string as output, and yet not infringe copyright in the AutoCAD lock?

On Northrop J’s reasoning, the answers would seem to be “No” to both questions.

If that is right, then the intellectual and philosophical underpinnings of copyright law have been profoundly altered by the Judgment.

FURTHER ISSUE

There was a further issue raised at trial, on which Autodesk failed. Autodesk advanced arguments based on the proposition that running a program is an act comprised within the copyright. In essence, the argument is as follows:

(a) The only function of Autokey is to enable AutoCAD to run.
(b) Running AutoCAD involves uploading it from hard disk to RAM.
(c) RAM is a “form of storage from which the program can be reproduced”.
(d) Copying a program into RAM for the purpose of running it involves a reproduction of the program in a material form (i.e. RAM).
(e) Users of AutoCAD are not licensed to run AutoCAD otherwise than with the AutoCAD lock.
(f) Supply of the Autokey lock amounts therefore to authorising the doing of an act comprised in the copyright.

In part, the argument turned on the particular facts of the case, namely what was the nature of the licence which users acquire.
But the point is of general interest. The terms of standard licence agreements for computer programs often appear to assume that running the program is something which cannot be done without the permission of the copyright owner. There are several things to be said against this. Firstly, it depends on the proposition that RAM is a “form of storage from which the program can be reproduced”; if not, then reproducing the program in RAM does not involve reproducing it in a material form and is thus not an act comprised within the copyright. The fact is, that in general a program cannot be reproduced from RAM when it has been uploaded for running.

Secondly, it would mean that computer programs enjoy a protection of a sort which is qualitatively different from that enjoyed by other works. Generally speaking, it is not an infringement of copyright in a work to do the thing for which the work is primarily created. Thus, it is not an infringement of copyright in a book to read the book. It is not an infringement of copyright in a play or a musical work to perform the work (unless the performance is in public). The copyright in a film is not infringed by watching the film.

The purpose of a computer program is to run it. If Autodesk’s argument is right, the very act of running a program would be an infringement of copyright unless done with the licence of the copyright owner. It would follow from that, that copyright in a program is infringed if a genuine licensed copy were run by a person other than the licensee. So, the licensee would not be permitted to allow a friend to run the program; presumably, it would not be possible to give a genuine licensed copy of a program to another as a present, since the recipient would not have a licence: a licence is not inevitably transferable.

THE APPEAL

Perhaps not surprisingly, the Respondents appealed to the Full Federal Court. The appeal came on for hearing on the 13th March, 1990. The bench comprised Justices Lockhart, Sheppard and Beaumont. Each of those had been involved in the litigation between Apple and Computer Edge. Beaumont J. was the Trial Judge in that case, and Sheppard and Lockhart JJ. comprised part of the appeal court in that case. On 14 September 1990, all three Judges allowed the appeal.

As might be expected, the judges considered in great detail the terms of the 1984 amendments, and considered the difficulties associated with protecting computer software by means of copyright. The court acknowledged that function is of critical importance in determining whether a program exists. But they held that copyright does not protect function.

The Full Court held that neither lock was or contained a computer program. They found that the AutoCAD lock together with Widget.C constituted a program. They held that the two locks employed different algorithms which were implemented differently. Hence Kelly’s lock (AutoKey) did not infringe Autodesk’s copyright in the program which comprised the AutoCAD lock together with Widget.C.

On the third issue, the Full Court found it unnecessary to decide whether running a program constitutes or necessarily involves a “reproduction” of the program. Lockhart and Beaumont JJ. left the question open; Sheppard J. was inclined to think it did not.

Autodesk has now brought an appeal against the decision of the Full Federal Court. The High Court will hear that appeal in April 1991.

At the time of writing, the Copyright Law Review Committee, acting on a reference by the Acting Attorney-General, is considering what form and scope of protection is appropriate for computer programs. The issues debated in the Autodesk case provide a valuable real life example of the difficulties involved in solving the problem of software protection by the “quick fix” approach adopted in 1984. Clearly, what is needed is a careful consideration of the nature and scope of protection needed for software, and the legislative creation of a form of protection specifically tailored for the purpose.

BIOGRAPHICAL NOTES

Julian Burnside attended Monash University and graduated Bachelor of Economics (1971) and Bachelor of Laws (1972). He was admitted to practice in 1974 and went to the Bar in 1975. He was subsequently admitted to the Bar in New South Wales, Tasmania, Western Australia and A.C.T. He was appointed Queen’s Counsel in 1989. He founded the Victorian Society for Computers and Law in 1980 and was its inaugural President. He is currently Chairman of the Commercial Law Association (Victoria) and President of the Melbourne Musicians Society. His interests include classical music, carpentry, photography, wine and reading.
I am willing, in my indicated area of expertise, to act as:

☐ Reviewer of books
☐ Referee for submitted papers

If you are willing to act in either capacity for the journal, please fill in the details below, then cut this page from this issue, and return it to the undersigned as soon as possible.

NAME & POSTAL ADDRESS

PRESENT POSITION

TELEPHONE
Home (0 ) Work (0 ) Fax (0 )

ACSNet address for electronic mail (if applicable):

On the reverse side of this form is a list of subject categories. Please use a highlighting pen to delineate the subject area(s) where your interests and experience lie. Then, for up to five categories representing your major interests, please give more qualifying detail, as in the following example:

1 d.3 Pascal standardisation; Modula-2; COBOL extensions
2 h.4.1 on-line banking systems; Hogan software package

Your five major Interests:
1
2
3
4
5

Please list other items (qualifications, work experience, recent publications, etc.) that are relevant to the subject categories that you have specified.

Other Comments:

Signed: Date:

When completed, please return this form to
Dr C.J. Andrews, ACJ Book Reviews Editor,

148 Jesmond Road, Indooroopilly, Qld. 4068.
<table>
<thead>
<tr>
<th>Hardware</th>
<th>Theory of Computation</th>
<th>Mathematics of Computing</th>
<th>Information Systems</th>
<th>Computing Methodologies</th>
<th>Computing Milieux</th>
</tr>
</thead>
</table>
| b 1. control structures and microprogramming  
 b 1.2 arithmetic & logic structures  
 b 3. memory structures  
 b 4. input/output & data communications  
 b 5. register-transfer-level implementation  
 b 6. logic design  
 b 7. integrated circuits | f 1. computation by abstract devices  
 f 2. analysis of algorithms & problem complexity  
 f 2.1 numerical algorithms & problems  
 f 2.2 nonnumerical algorithms & problems | g 1. numerical analysis  
 g 2. discrete mathematics  
 g 2.1 combinatorics  
 g 2.2 graph theory  
 g 3. probability & statistics  
 g 4. mathematical software | h 1. models & principles  
 h 1.1 systems & information theory  
 h 1.2 user/machine systems  
 h 2. database management  
 h 2.1 logical design  
 h 2.2 physical design  
 h 2.3 languages  
 h 2.4 systems  
 h 2.5 heterogeneous databases  
 h 2.6 database machines  
 h 2.7 database administration  
 h 2.8 database applications  
 h 3. information storage & retrieval  
 h 3.1 content analysis & indexing  
 h 3.2 information storage  
 h 3.3 information search & retrieval  
 h 3.4 systems & software  
 h 3.5 on-line information services  
 h 3.6 library automation  
 h 4. information systems applications  
 h 4.1 office automation  
 h 4.2 types of systems  
 h 4.3 communications applications | i 1. algebraic manipulation  
 i 2. artificial intelligence  
 i 2.1 applications & expert systems  
 i 2.2 automatic programming  
 i 2.3 deduction & theorem proving  
 i 2.4 knowledge representation formalisms & methods  
 i 2.5 programming languages & software  
 i 2.6 learning  
 i 2.7 natural language processing  
 i 2.8 problem solving, control methods & search  
 i 2.9 robotics  
 i 2.10 vision & scene understanding  
 i 3. computer graphics  
 i 3.1 hardware architecture  
 i 3.2 graphics systems  
 i 3.3 picture mage generation  
 i 3.4 graphics utilities  
 i 3.5 computational geometry & object modelling  
 i 3.6 methodology & techniques  
 i 3.7 three-dimensional graphics & realism  
 i 4. image processing  
 i 5. pattern recognition  
 i 5.3 clustering  
 i 6. simulation & modelling  
 i 6.1 simulation theory  
 | j 1. administrative data processing  
 j 2. physical science & engineering  
 j 3. life & medical sciences  
 j 4. social & behavioural sciences  
 j 5. arts & humanities  
 j 6. computer-aided engineering  
 j 7. computers in other systems | k 1. the computer industry  
 k 2. history of computing  
 k 3. computers & education  
 k 3.1 computer uses in education  
 k 3.2 computer & information science education  
 k 4. computers & society  
 k 4.1 public policy issues  
 k 4.2 social issues  
 k 4.3. organizational impacts  
 k 5. legal aspects of computing  
 k 5.1 software protection  
 k 5.2 governmental issues  
 k 6. management of computing and information systems  
 k 6.1 project and people management  
 k 6.2 installation management  
 k 6.3 software management  
 k 6.4 system management  
 k 7. the computing profession  
 k 8. personal computing |

This book is part of a series, Professional and Industrial Computing. Others in the series are:
- Veryard: Pragmatic Data Analysis
- Taylor: PICK for Users
- Jennings: Practical Data Communications
- Walsh: Computer Users Data Book
- Purser: Computers and Telecommunications Networks

I thought it worthwhile listing the rest of the series as, if they are written as concisely and practically as this one, then it is a good series.

Mind you, if you find this book invaluable then your installation must be in real trouble. The book I think would benefit students mostly from a learning point of view, professionals to review their attitude toward standards, and users to understand better the necessity of having them.

Documentation has always been the bane of professionals and standards criticised as being in the way of creativity. That is a lot of rot. This book gives a good outline and indeed checklists of what should be in a standards document. It is easy to read in total and the way it is set out makes it a handy reference guide.

I have, however, two criticisms. I think the second part of the title is a bit of a misnomer, i.e. 'for Data Processing Management'. ‘A Product Guide’ would have been much better. Secondly, $44.00 seems a bit high for the quality of the publication, although the quality of the information probably warrants it. David J. Riches


Presented with a problem to solve, it is seldom that a person builds a plan from first principles. The human mind draws on previous experiences, checks them against current objectives, makes modifications, re-checks and re-modifies, until a suitable plan is arrived at. Professions such as law, business and medicine have refined this process so that previous examples are extensively documented and indexed. The teaching of these disciplines is based largely on previous cases.

Case Based Planning sets out to show how an expert system can mimic this approach to reasoning. In one short introductory chapter, Kristian Hammond presents the theory of planning as an exercise in learning. Given a set of goals, the case-based ‘planner’ (computer program) considers how they will interact; anticipates any previously recorded failure and searches for a known plan which satisfies the goals and solves the ‘predicted’ problem. Any failures which occur are used to repair the flawed plan and to amend its knowledge base, so that the same failure is not repeated.

Indexing of plans, then, is based on the goals they satisfy and the problems they avoid. ‘Case-based planning’ differs most significantly from ‘rule-based systems’ in its ability to derive new plans from past successes and past failures. Although case-based planning is still in its formative stages, this introduction mounts a powerful argument for its use.

The remainder of this book uses the expert system ‘CHEF’ to illustrate the principles of case-based planning applied to the world of Szechwan (Chinese) cooking. CHEF began with 10 recipes and by the time of publication had created 21 more, which, after simulation, it deemed edible. Along the way the need is established for a strict vocabulary with which to examine and describe the causes of failure and a set of inference rules against which to check each step of the plan.

Generally the various facets of the planner are well illustrated, if somewhat repetitively. An overview diagram of the component parts of the CHEF system, with appendices detailing the rules embodied in it, would have provided the reader with an easier introduction to the detailed chapters. Overall, however, the book achieves its purpose: one is left with the strong impression that the planning function can be modelled effectively using this technique. Despite a determined effort to remember throughout that CHEF was merely illustrative of the planning domain, by the end of the book there was a distinct feeling of over-indulgence on strawberry souffle, duck pasta and other gastronomic delights.

Case Based Planning is the first in a projected series of texts on ‘Perspectives in Artificial Intelligence’. While it could not be considered a primer on artificial intelligence or expert systems, it is certainly a book which provokes the reader to pursue the idea of mechanism of cerebral activities. It is perhaps an unusual selection to introduce a series, but the choice is a stimulating one.

Roger Price
State Electricity Commission of Victoria


Until recently, Computer Graphics research has almost completely ignored the Software Engineer’s call to mathematical formalism, professing the excitement of quick (and sometimes dirty) methods for producing nice pictures. The cost of this ignorance is high; it has led to graphics textbooks filled with vague specifications and incorrect algorithms, and to graphics standards which are tied to the software concepts of the 1960s.

Fiume’s book reports one of a small number of recent attempts to redress the problem. It is ambitious, providing a formal basis for a great deal of the graphics pipeline between scene specification and screen image.

Chapter 1 introduces an elegant specification method for scenes, and transformations on these scenes. Rather than adopting one of the currently popular formal specification languages, Fiume uses standard mathematical notation.

Chapter 2 and 3 treat visibility and rendering problems respectively. The framework described for rendering is powerful enough to capture all known rendering techniques; unfortunately, it is mathematically complex.

An approach to bitmapped graphics, including a formal treatment of rasters and an analysis of line rasterization, is described in the next chapter. Illumination models, including some interesting mathematical results on the complexity of ray-tracing, are presented in the last two chapters.

The book would make an interesting case study for students in a course on formal methods, and could be used to add weight to a computer graphics course. Although it may not be the last word in the formalization of graphics, it represents one of the first steps toward the maturation of the field.

Peter Eagles
University of Queensland


Society as a whole is increasingly dependent on computers. It is therefore normative for each generation to extend its use of computers increasingly. Thus usage of computers by many people can be regarded as ‘normal’ or ‘usual’. Some people give computing an excessive salience in their lives. Margaret Shotton demonstrates that some individuals have an engagement with computers that is quite different qualitatively and quantitatively from other social computer users. She writes that the computer user’s ‘need to control his (her) environment is neither neurotic nor pathological but provides an admirable means for coping for those who may previously have felt inadequately fulfilled’ (pg 168).

The author statistically differentiated three separate psychological groups of computer dependent persons -

1. Explorers: Those who are dependent on exploring the system and educating themselves. The computer becomes an end in itself.
2. Workers: Those who program toward an end product.
and who also use commercial products, for specific purposes. The computer is a means to an end.

3. Networkers: Those who use computers to access billboards, data bases and to play network games. The computer is again a means to an end.

Some computer dependent persons find that computing not only stimulates and educates and gets the job done, but also serves as source of common interest with other computer users and enriches their social lives. For the seriously computer dependent person the computer actually reduces his socialization because the computer is for him more satisfying (pg 212).

The author's use of the concept of dependence and addiction is analogous to the paradigm used for drug and alcohol dependence by psychiatrists. The inability to abstain from the temptation to 'compute' and the inability to control the duration of computing are exactly analogous concepts to those used in drug dependence.

Computer dependency is a new concept so the natural histories of the course of the three computer addictions are not yet known. One hopes that the author will follow up the same people at five year intervals to determine what the emergent positive and negative consequences of computer dependency will be.

This book is an academically sound treatise, is easy to read and comprehend and is filled-with facts. It is definitely recommended to educators, psychologists, psychiatrists and computer scientists and computer enthusiasts.

**Dr. Alan H Freed**

**Psychiatrist, Ipswich, Qld.**

---

**CHABRIS, C.F. (1989): Artificial Intelligence & Turbo C, Dow Jones-Irwin, 385pp., $56.50.**

Artificial Intelligence (AI) is usually associated with symbol manipulation and consequently with programming languages such as Prolog and LISP. Why should readers be interested in this book which associates AI and the programming language C, and specifically Turbo C? An answer to this question is provided on p 19 where we learn that 'One function of this book is to provide a total resource for the Turbo C programmer who is interested in learning about artificial intelligence'.

I found this book easy to read and entertaining. It covers the basic aspects of AI such as knowledge representation, heuristic search, game playing, expert systems and natural language processing. The first two chapters cover history and philosophical issues. There are many programs and demonstrations. The examples are often refreshingly different from those often seen in other introductory books on AI. There are many references to the literature, and my overall impression is that Chabris is not just an enthusiastic C programmer.

Although the book contains 385 pages there are several long appendices extending over 157 pages. The largest of these contains source listings of C programs which are introduced in the main chapters. This would be more useful if it were available on disk, but there appears to be no mention of this option in the book. Three other appendices contain references to publications, organisations and business associated with AI. Although some of this will date fairly quickly the extensive bibliography appears to be very useful. One other appendix offers a glossary of AI terminology.

A book containing a number of AI-related algorithms can take the mystery out of AI. AI algorithms have to be written in some programming language and are compiled in the usual way. Unfortunately they are usually very large. This is a real problem since presenting small subsets of AI algorithms can make them trivial, while large algorithms can be difficult to describe. In this book Chabris has chosen to present a number of small or cut-down algorithms. For example a description of a simple production-system implementation appears in the chapter on expert systems. This illustrates the concepts of forward and backward chaining but is barely sufficient to build an expert system.

The chapter on heuristic search is weak. Several example algorithms are presented which do not match up with examples of how they are used. Also the graph search algorithm A* is presented as a tree search. Some properties of real A* are justified which do not hold with the implemented version. This seems to be an instance where the AI algorithms are simplified for ready exposition beyond a point which is sensible.

In summary the title of this book is appropriate for the contents. Those people with minimal experience of Turbo C can read the book without disadvantage. However taking the price into consideration I am not able to recommend the book unreservedly.

**Phil Collier**

**University of Tasmania**

---


This book is likely to be useful to anyone concerned with computerization of social welfare organizations, particularly if there is some resistance to the idea of computerization. 'Computerized information systems can be destructive or constructive for organizations and staff'. If staff won't use computers, then they are of little value. Thus it is important, before introducing computers, to have free discussion with all staff and take careful note of what they say. The other group who would find this book worth having are libraries who need reference works to help answer such questions as whether computers put human values at risk: confidentiality, discretion, initiative and personal relations. Work published up to about 1980 suggests dangers in computerization, but this 'is simply not borne out by the empirical work completed since 1980'.

The author is reassuring with some of his findings. Computer use and systems were designed primarily as management tools, and computerization was never as bad as some people feared it was going to be. Loss of direction, the threat to privacy, de-individualization, appear to be relatively unimportant to staff. Job security and the use of computers to monitor staff turn out to be non-issues. On the other hand, computerization was seen as giving staff more control over their jobs, and also increased productivity.

The book uses two descriptions of types of social work organizations: 'people-processing' and 'people-changing'. People-processing tries to change clients by giving them a public status and relocating them in a new set of social circumstances; people-changing tries to alter the behaviour of clients through various modification and treatment technologies. However, so far as I could see, it made little difference to the acceptance or use of computers which of these types of organizations was involved. 'The computer is a toothless enemy, at least so far'.

I found it difficult to read many of the research papers, especially when they had multidimensional polysylablification. However, there is a passage (pp 182-3) starting: 'At first blush the response of staff ...' and concluding to 'Whether the computer, swallowed whole like a stone, will pass through the system, leaving it unchanged or instead disrupt its functioning in serious ways, remains to be seen', which is so clear and forceful, that I'd like to know who wrote it.

A couple of technical points, which I think would help the non-technical reader:

1. There are good summaries in each chapter, but if they were put at the start of each chapter, and emphasised (NLQ rather than draft) it would clarify the situation, and make the reading easier.
2. The final two pages, though not called a 'Summary', are important and would justify the same treatment.
3. I am not happy about the whole statistical approach to this problem: if a question (A7) is 'Computer use forces client data into artificial categories' and 50% agree and 43% disagree (N = 61), the first question I would like to ask is 'Did they understand the question?'

Nevertheless, I think the findings are sufficiently valid to make this book worth keeping as a reference.

**Chester Wilson**

**Ipswich Qld.**
NEWS BRIEFS

“News Briefs from the Computer Worlds” is a regular feature which covers local and overseas developments in the computer industry including new products, and other topical events of interest.

APBC OFFERS ELIGIBLE TRAINING IN MARKETING AND PUBLIC RELATIONS

In response to the Training Guarantee Act the Australian Progressive Business College (APBC) has carried out research with marketing and communication managers of major Australian companies to identify the most important training requirements in the current economic climate. Not surprisingly, most respondents stressed the need for skills in strategic planning.

This research has led to the introduction of one year part-time courses in Marketing and Public Relations to provide participants with practical skills in developing effective product or promotion strategies. These courses offer hands-on continuing education for people whose present jobs include either marketing or public relations functions. The fee for a complete course is $2965 which is tax deductible and eligible for the training guarantee.

The course on Marketing places particular emphasis on issues common to the computer industry — namely the marketing of hardware, software and services in the various horizontal and vertical markets associated with the computer industry.

The Training Guarantee Act requires organisations which pay more than $200,000 in wages to spend one per cent of their total wages bill on job-related training... or they must pay this amount to the Taxation Office and lose the tax deduction.

The one year Marketing course is offered in modules which include strategic marketing and planning, market research, sales management and promotions management. On completing the course participants will receive a Diploma of Marketing. The APBC is accredited by the Australian Marketing Institute.

The Public Relations course includes public relations fundamentals and case studies, publicity, media relations and media production. The APBC is the only training body endorsed and supported by the Public Relations Institute of Australia (NSW).

NEW PERTH UNIVERSITY

On 5 December 1990, legislation passed through Parliament to change the designation of the Western Australian College of Advanced Education (WACAE) and establish the institution as the Edith Cowan University.

From 1 January 1991, the Edith Cowan University has assumed full responsibilities of the Western Australian College of Advanced Education.

The College was formed in 1982 through the amalgamation of four colleges located at Claremont, Nedlands, Mount Lawley and Churchlands.

The College has its roots in the beginning of this century with the establishment of the Claremont Teachers College in 1902, making it the oldest tertiary institution in Perth.

In the past 89 years, the institution has grown to enrol some 15,000 students in more than 160 undergraduate programmes and has diversified to include substantial postgraduate and research elements.

Today, the Edith Cowan University is the second largest university in Western Australia and is ranked 13th out of some 34 Australian universities, educating 30 per cent of the State’s university students.

Thirty five per cent of the country university students enrol at the Cowan University which also has the biggest enrolment of mature age students in Australia.

Additionally, Edith Cowan University has some 670 full-time paying students mainly from Malaysia, Singapore, Hong Kong, Indonesia, Thailand, Japan, Taiwan, the Philippines, South Korea, The People’s Republic of China and The Seychelles.

Edith Cowan University degree courses carry full academic status in Australia through its accreditation and listing in the Register of Australian Tertiary Education (RATE), an official government body representing all government education ministries in Australia.

All students undertaking studies at Edith Cowan University enrol in fully approved and accredited academic programmes that have strong professional recognition in Australia and overseas.

The decision to designate the College a university has crowned the continued growth of the College since its formation and was also a culmination of the high standards and commitment shown by staff and students over a long period of time.

AUSTRALIAN BUSINESS IN EUROPE

As the European single market rapidly approaches, it is increasingly important for Australian companies to identify and capitalise on investment and export opportunities. The challenge is enormous: the 17 member nations of the European Community (EC) will constitute a single market containing 340 million people. And since an EC-commissioned report estimates that by 1998 there will be up to five million extra jobs created, Europe’s consumer demand is expected to grow. There will be no tariffs between the EC nations and mutual standards will be adopted in every business arena.

Jim Hayman, president of the recently formed Australia branch of the British-based Australian Business in Europe (ABIE) says: “We’re looking at the biggest market in the world with standardised regulations and documentation for importing, exporting, insurance and banking. Get into one country and you’re into the whole market in a single lick”. However it’s tough getting in, and succeeding, and that’s the main reason ABIE was formed in Britain in 1975 — to provide a focal point for the exchange of information and experience which can help members improve their success rate in business between Europe and Australia.

ABIE now has over 200 members across Australia and 700 internationally, with members from corporate giants such as CSR Ltd and Western Mining, accounting and law firms and banks, to small businesses, consultancies and individuals.

Speaking at an ABIE lunch in Melbourne, Federal Minister for Trade Negotiations, Dr Neal Blewett said: “ABIE shows great promise because of its network across Europe and now in Australia, and because of its diverse membership, which will allow considerable sharing of expertise.”

“ABIE brings together the major players in Australian banks and companies which operate in Europe; specialist firms in accounting, insurance and stockbroking; representatives of state and federal government departments and of consulates — general and foreign chambers of commerce; as well as companies which are developing niche markets in Europe in products ranging from measuring equipment to ti-tree oil.”

Dr Blewett said the Australian government will “be looking to ABIE to help us co-ordinate our efforts in Europe,” and announced at the meeting the formation of a new high-level business advisory group on Europe, to be chaired by Sir Peter Abeles.

Appointments of other members will be made by the government and are expected to be announced shortly.

Europe’s importance to Australia must not be underrated. Addressing another ABIE function, His Excellency Mr Ove Juul Jorgensen, Head of the Delegation of the Commission of the European Community to Australia and New Zealand, pointed out the following: “Total EC-Australia trade amounted to $A 18 billion in 1989. The EC has been Australia’s largest supplier of imports and second biggest export market for a good many years. Moreover about one-third of total foreign investment in Australia is derived from the community.”

However, many Australian businesses may be underestimating the competition they will face in the single European market, according to “Europe 1992: Taking Action”, a survey prepared by ABIE and The Australian High Commission in London.

The survey of 104 Australian firms operating in or exporting to the EC, found that these companies view the EC as the most important market outside Australasia and Southeast Asia, and the most important investment location.

Protectionist attitudes against non-EC countries was the greatest concern of respondents. Releasing the survey at an ABIE function, Dr Blewett said it was worrying that only 24 per cent of the respondents lobbied the European Commission, and the survey urged Australian companies to unite.

ABIE helps businesses initiate and sustain European enterprises. In addition to the benefits of the network of information, and people with common interests, members receive an international directory classified by occupation, regular newsletters with information about doing business in Europe, and invitations to functions to exchange news and hear guest speakers well versed in European affairs.
NEWS BRIEFS (Continued)

NEC AUSTRALIA WINS $1.5 MILLION CONTRACT WITH UTS

NEC Australia has won a $1.5 million contract with the University of Technology, Sydney to supply a complete new digital communications network.

The letter of intent was signed in Sydney by UTS Vice-Chancellor Professor Gus Guthrie and NEC Australia's General Manager of the Business Communications Division Brian Hutchison.

The new network of NEC NEAX 2400 PABXs will provide voice, data and compressed video transmission facilities. All inter-campus communications, including facsimile, will be internal, with every member of staff having an individual extension.

The new system will serve three sites which constitute the University's City campus, three sites on the St Leonards campus, as well as the Balmain and Ku-ring-gai campuses.

According to Professor Guthrie this is the outcome of almost two years planning and development of a communications policy which would support the University's mission.

AUSSAT AND IBM SIGN $3.8 MILLION CONTRACT

IBM's Interactive Satellite Education Network (ISEN) has been so successful over the last year that the company recently signed a new $3.8 million contract with AUSSAT.

The three year contract will allow expansion of IBM's ISEN network, which provides training and educational services to the company's staff, customers and industry. Classrooms have already been established in Canberra, Adelaide, Melbourne, Brisbane and Perth which receive, via satellite, interactive broadcasts originating in Sydney.

Using AUSSAT's earth station in Auckland and in Wellington (to be commissioned in June 1991) IBM will also expand its network to cover New Zealand for the first time using AUSSAT's Taslink service.

ISEN is used for many applications, including meetings, product launches, seminars, remote trouble shooting and, of course, training for IBM staff and customers. The network is also available for hire.

"IBM has been very satisfied with the results of AUSSAT-based ISEN. It has been widely accepted throughout the workplace and we are now extending our training services to industry at large," said Mark Riddell, Education Manager, IBM Australia/New Zealand.

"This new contract reinforces the commitment IBM has to AUSSAT, and the use of satellite technology," he said.

"Our experience has demonstrated significant cost savings and has added a degree of flexibility to our operations that cannot otherwise be achieved," Mr Riddell added.

AUSSAT distributes the dedicated 2 megabit links across Australia and the Tasman using digitally compressed signals, which allow two way voice and data, and one way video.

STATE BANK OF NEW SOUTH WALES LAUNCHES INNOVATIVE FOREIGN EXCHANGE TRADING SYSTEM

State Bank of New South Wales Limited in conjunction with Logica has successfully implemented an innovative foreign exchange trading system that introduces unique dealer alarm functions and significantly faster responses for spot trading.

Developed with the specific needs of the dealer in mind, the system is the first module of State Trader, the Bank's integrated trading system. It maximises the power and sophistication of workstation technology, offering immediate access to market feeds and back office systems, combined with a customised, lucid graphic interface.

High performance has been achieved through the use of Teknekron Trading System (TTS), an information distribution platform at the heart of the Bank's new system.

Supplied by Logica, TTS has access to data for a variety of disparate sources including market feeds and back office systems. It customises this data both for viewing on a dealer's screen and for processing in real time by application software. A high level of resilience is also provided, as TTS balances loads between resources so that system failure and recovery does not affect dealing room communications.

At the Bank's Grosvenor Place dealing rooms in Sydney, State Trader runs on Sun Microsystems SPARC stations, connected on an Ethernet. The X windows graphical user interface standard is used to give features a common appearance, allowing other applications to be added at later stages.

INTEGRATED PC/GPS POSITION

Revolutionary in its concept, turning your personal computer into a self-contained global positioning system is now 'on the cards'.

Navstar Ltd, a member of the Peek plc group and a British designer and manufacturer of GPS (Global Positioning System) Receivers for ten years, has launched a simple expansion card containing a GPS Receiver for slotting into any IBM XT or AT compatible PC.

The Dynapulse 200M is a clinical accuracy blood pressure/pulse rate measurement and recording system. The device connects to any standard PC via a serial port and provides full on-screen instructions as to use. In addition to measuring blood pressure and pulse rate, the Dynapulse 200M will store the results in a database and allow a history to be built up over a period of time. It is ideally suited for corporate and home use. For further information contact Microgen Computers on (043) 34 1544 or Fax (043) 34 4132.