LEGAL AND TECHNOLOGICAL PROTECTION OF COMPUTER SOFTWARE

Simon Mays Elsom

Doctor of Philosophy

The University of Aston in Birmingham

October 1983

ACKNOWLEDGEMENTS

My greatest debt is to my supervisor, Professor Ernest Braun, whose constant encouragement made this thesis possible.

Many other people provided valuable help at various stages of the research. While it is not possible to list them all, the following were particularly important:

- John Collins, for his valuable comments and criticisms on a draft of Chapter 4, and
- Peter Prescott, for his valuable comments and criticisms on a draft of Chapter 2.

I would also like to thank the Science and Engineering Research Council for funding the research.

Last, but by no means least, I owe a great debt to my wife, Pauline, both for her patience in putting up with a research student over the last couple of years, and undertaking the daunting task of typing up my work.

THE UNIVERSITY OF ASTON IN BIRMINGHAM

Legal and Technological Protection of Computer Software

Simon Mays Elsom, Doctor of Philosophy, 1983

SUMMARY

This thesis is concerned with the effects intellectual property, and allied, rights have on the development of inventions and innovations in a particular technology.

The scope of legal protection for computer software has, for many years, been uncertain. Although this has little practical consequence for software firms developing programs for small, specialist markets, it has caused serious problems for firms supplying programs to mass markets.

The thesis can be characterised as one large case study comprising three parts. First, an analysis of the applicability and scope of protection provided by intellectual property, and allied, rights to software; secondly, the results of a comprehensive survey, concerning software protection, of firms in the UK computing services industry; and thirdly, a description of various technological protection measures that have been written into programs.

Computer programs, together with other items of software, are probably copyrightable. However, the subsistence of copyright for computer program works and the scope of the program copyright owner's exclusive rights, have yet to be considered in the High Court. A residual uncertainty exists, and as a result software firms are deterred from enforcing their rights in court. Many participants in the survey, and particularly those from microcomputer software firms, had reservations on the effectiveness of copyright, and other legal rights, as modes of protection for their programs. These firms have developed alternative techological forms of protection. Although a variety of different technological measures are used, many are designed to prevent those activities that effective law would, in most instances, deter.

Previous studies concerning the roles intellectual property rights play in the process of technological innovation have assumed that they have no effect on the inventions and innovations developed. However, for computer software, a perceived lack of effective legal protection has stimulated the development of, alternative, technological forms of protection.

KEYWORDS : Innovation, Software, Protection, Law.

C O N T E N T S

LIST OF ABBREVIATIONS

ADAPSO	Association of Data Processing Service Organisations
CCPA	Court of Customs and Patent Appeals
CONTU	National Commission on New Technological Uses of Copyrighted Works
CPU	Central Processing Unit
CRA	Computer Retailers Association
CSA	Computing Services Association
DES	Data Encryption Standard
DOS	Disk Operating Systems Software
PROM	Programmable Read-Only-Memory
EPROM	Erasable PROM
EPC	European Patent Convention
FSR	Fleet Street Law Reports
RAM	Random-Access-Memory
ROM	Read-Only-Memory
RPC	Reports of Patent Cases
TRM	Tamper Resistant Module
USC	United States Code

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

THE PROTECTION OF TECHNOLOGICAL INNOVATIONS

Technological change has, for a long time, been regarded as having a central role in economic progress.[1] However, only in the last 10-15 years or so, has the process of technological innovation been considered in detail. This process is distinct from the steps inherent in creating an invention or work. Indeed, the invention or work can be the beginning of a long period of research and development before it is transformed into a new, and proven, commercial product, a technological innovation. The process of technological innovation comprises all the steps, or stages, from initial conception to a final, and proven, commercial product, method etcetera. Thus in his book "The Economics of Industrial Innovation",[2] Freeman distinguishes, at the outset, between an 'invention' and an 'innovation' as follows:

> "An invention is an idea, a sketch or model for a new or improved device, product, process or system ... An innovation in the economic sense is accomplished only with the first commercial transaction involving the new product, process, system or device ..."[3]

Furthermore, he, amongst others, recognised that the process of technological innovation requires not only an inventor, the person who creates the invention or work as the case may be, but also an innovator, product champion, entrepreneur, publisher or investor. He transforms it into an innovation, for he organises,

and perhaps even finances, its development.

Technological innovation can be characterised as a process of diffusion, or sharing, of knowledge. The inventor divulges details of his creation to an innovator in order to attract investment for its development.[4] The innovator, in turn, divulges details of the resulting product, process, system or device as the case may be, to others who may wish to use it, in order to obtain an economic return from his investment. Publication is an essential element in the process of technological innovation. However publication has its risks, for an innovation has a market value. In the technology of interest here, the development of an innovation is expensive, requiring highly skilled manpower, but once achieved, the innovation can easily, and cheaply, be reproduced. The innovator risks one, or more, of his customers copying the innovation and competing for his markets. In extreme circumstances, the innovator may be deprived of an adequate return from his investment. The invention also has a market value, and the inventor likewise risks an unscrupulous entrepeneur taking advantage of the trust placed in him.

Although the probability of unauthorised development of some inventions, and the unauthorised reproduction of some innovations, can be small, it has long been feared that without effective protection, inventors may be deterred from divulging details of their creations, and innovators, in turn, deterred from investing in them. In either case, the process of technological innovation breaks down. Thus, for over 100 years, inventors and innovators have obtained protection, in law, in the form of 'intellectual

property'.[5]

An object of all intellectual property rights is that inventors are encouraged to publish their work, rather than keep it secret, secure in the knowledge that should their rights be infringed they can obtain redress in court. Intellectual property rights also encourage the innovator to invest in the work.[6]

Intellectual property has evolved into a number of distinct rights, including Patents and Copyright. Each protects a particular type, or types, of information. The intellectual property owner has, in English law, various economic rights in the information, or its expression, recognised by the courts.*

Patents have been accredited a number of functions for the inventions they protect, including:

(1) they allow the inventor/innovator an economic return on the effort expended in devising his invention/innovation;[8] and

(2) they provide bargaining counters in exchanges of technical information between companies.[9]

It has also been argued that patent protection encourages the disclosure of new and improved techniques.[10]

Copyright has been accredited a number of similar functions

---

* Some commentators distinguish between 'industrial property rights', patents, trade marks and industrial designs used to protect inventions, and 'intellectual property rights' such as copyright used to protect cultural works.[7] It will become apparent that in the technology of interest here, this distinction is unworkable. Thus the term 'intellectual property' as used herein includes so-called industrial property rights and copyright.

for the works it protects, and particularly:

(1) it attracts the investment required to produce the work in the first place;[11]

(2) it ensures that the author benefits from the fruits of his labour;[12]

(3) it ensures that the author's expended skill and/or labour is protected.[13]

Patents and copyright have a central role in the process of technological innovation, for they provide protection for the inventor and innovator in respect of their efforts, once they have been published. It is surprising therefore that the role and effect of intellectual property rights in the process of technological innovation have not previously been considered.

Most of the previous studies into the process of technological innovation have assumed that the inventions, works, and innovations of interest are, at least, capable of intellectual property protection.[14] In one sense it is understandable that these studies have concentrated on easily-protected creations, for the principal source of data is patent specifications.[15] However, this source is far from ideal for it only contains details of those inventions and innovations that are both patentable, and have been patented. It is, in effect, an incomplete record of certain types of invention and innovation. It does not, for example, contain details of unpatentable inventions, or those patentable inventions which, for one reason or another, have not been patented. In his book "Invention and Economic Growth",[16] Schmookler was careful to point out the limitations of

patent data in his study on technological innovation, but he nevertheless assumed that most valuable inventions and innovations were both patentable, and that they had been patented.[17]

This thesis can be characterised as a case study of computer programs, a technology that sits uneasily between the traditional subject matter of patentable inventions, and the traditional subject matter of copyrightable works.

A computer program is designed to condition, or otherwise control the operation of electronic circuits comprising a computer, for some end. It can, for example be a collection of integrated circuits, an engineering embodiment perhaps capable of patent protection. It can also be written down in a form that may attract copyright protection.

It is the dual nature of a computer program that, in theory at least, creates a protection problem. There is considerable uncertainty concerning both the copyrightability of engineering embodiments of a program, and the scope of the program copyright owner's exclusive rights. Also, it is uncertain whether programs, in any embodiment, are capable of patent protection. The scope of protection provided by these major intellectual property rights for computer programs has yet to be clarified.

One of the few published studies, on the effects of law in the process of technological innovation in the computer software industry, is contained in R.I. Miller's book "Legal Aspects of Technology Utilization".[18] He was not concerned with innovation in software per se, rather the 'commercial utilization'[19] of

software inventions. He was particularly interested in whether, or not, US intellectual property rights had an enhancing, or inhibiting, effect on the utilisation of these inventions.[20] As part of these studies, in 1973, he conducted a survey of the, then, small US Software Industry. This comprised a postal questionnaire to the 46 members of the US Association of Data Processing Organisations (ADAPSO).[21] The objectives of this survey were:

> "... to poll its membership on the types of legal protection used for software, the relative satisfaction with the available modes of protection, and whether legal barriers are ever instrumental in discouraging or preventing the development or marketing of software."[22]

From this survey and other data, he concluded that, in general:

> "The law has a negligible effect either as an incentive or as a barrier to the progress of an innovation from its reduction to practice until its commercial utilization."[23]

Miller considered the process of technological innovation, from initial invention to the diffusion of the resulting innovation in the market, to be interactive. According to his model, market acceptance of an innovation, in turn, led to new inventions, improvements in the product, and thus new innovations.[24] However, since he was concerned with the utilisation of inventions, he did not consider the effects, if any, intellectual property rights had on the resulting innovations. Rather, he implicitly assumed that whatever factors, or circumstances, caused the development of a particular innovation, its protection

was not one of them. Thus, while the existence, or otherwise, of effective intellectual property rights did not seemingly influence the decision to develop certain software inventions, a lack of effective protection could have influenced their development, and therefore also influenced the resulting innovations.

In 1977, the US National Commission on New Technological Uses of Copyrighted Works (CONTU) commissioned a survey of the US computer software industry[25] to determine, amongst other things, whether a lack of legal protection for software had inhibited software's development. On the basis of 116 replies to a postal questionnaire sent out to the, then, 308 members of ADAPSO, it was concluded:

> "The typical company relies largely upon its technological resourcefulness in a burgeoning industry. It is not particularly concerned with the protection of the software that it develops or purchases and, to the extent that it is, would prefer to rely upon physical, TECHNOLOGICAL, and contractual modes of protection rather than legal monopolies."[26] [Emphasis is mine]

Legal protection may not have inhibited development, but it certainly influenced software development. The 'technological' modes of protection referred to above, are the fruits of investment channeled into software protection. Unfortunately, since the researchers were not concerned with the effects of inadequate legal protection, they did not elaborate on what these 'technological modes of protection' comprised.

The hypothesis underlying this thesis is that a lack of

adequate, and easily enforceable, intellectual property rights for software has led to the development of, alternative, technological modes of protection. Thus, rather than causing greater secrecy within the computing services industry, a lack of effective protection had resulted in a technological 'fix', that is, innovations designed to prevent, or deter, those activities that, in most instances, effective intellectual property rights could deter. Particular reference will be made to protection innovations for microcomputer programs.

The thesis discusses aspects of the relationship between legal protection of software, and the development and use of protection innovations used by firms in the UK microcomputer software industry. The structure of the thesis is as follows: Chapter 1 briefly describes the different types of computer program, the structure of the UK microcomputer computing services industry, and the need for effective protection. Chapter 2 discusses UK and US copyright statutes, and the applicability and extent of copyright to software and relevant case law. Chapter 3 discusses the remaining types of applicable legal protection for software, and particularly UK and US patent statutes and relevant case law. It also includes a brief discussion of the action for breach of confidence, and its relevance to the protection of software in the UK. Chapter 4 presents the findings of a survey of 156, mainly UK, software companies. The survey collected data, and canvassed opinions from personnel within the industry, on software protection and related issues. Chapter 5

describes some technological methods of protection for microcomputer programs currently being used. Conclusions are contained in Chapter 6, which also describes possible future developments in the legal and technological protection of software.

CHAPTER 1

COMPUTER PROGRAMS AND THE NEED FOR PROTECTION

INTRODUCTION

The protection of computer programs has only recently been considered seriously by companies in the computing services industry. Until the boom in microcomputer sales in the late 1970's, interest in the applicability of intellectual property protection, such as Patents and Copyrights, to computer programs was mainly academic. At that time there were few reported instances in which program authors' rights had been abused. Furthermore, computer programs themselves had features which inherently provided a wide measure of protection for their suppliers. For example, many mainframe and minicomputer programs were tailored to one, or a small number of, end users, and therefore there existed, at best, a small market for bootlegged copies. In addition, many of these programs required regular post-sale maintenance and updating in order to be implemented effectively by the computer user. Thus software suppliers were in regular contact with their customers. Also, the utility of a bootlegged copy to an unauthorised end user was likely to be shortlived. Sooner or later he would be forced to approach the bona fide software supplier for updates should he require continued use of the program. The actual circumstances of his purchase would then, in all probability, be discovered. Thus, although the scope of intellectual property rights for software

suppliers in respect of their programs was, and is, uncertain, it was hardly ever a problem of any great consequence to most of them. They had adequate protection already, built into their products.

However, some programs currently being marketed do not have these, and other, inherent protection features. For some software suppliers the protection of their products has ceased to be a matter of no great consequence. The combination of a large market for certain popular microcomputer programs, the ease with which such programs can be copied, and the use of seemingly inadequate and inappropriate modes of legal protection, has created a black market of indeterminate size for copies of such programs. For some suppliers there now exists a major protection problem in respect of their programs.

This chapter includes a brief description of a computer program, the development and operation of a typical program, the various types of program, and some of the technological and commerical developments that have created a protection problem in their wake. This chapter is intended to provide a background within which to place the detail contained in subsequent ones.

The succeeding section describes computer programs. This is followed by a section briefly describing some of the landmarks in the evolution of the computing services industry. A third and final section describes, in greater detail, the program protection problem outlined above.

COMPUTER PROGRAMS

In 1975, a survey of 48 UK Software Firms included the question, "How would you define a computer program?" Thirty-seven firms replied, and 37 different definitions of a computer program were obtained.[1] Although 8 years have elapsed since that survey was undertaken, there is no evidence to suggest that, in the meantime, a universally accepted definition has been agreed within the computing services industry.

In his book "Getting Acquainted with Microcomputers",[2] Frenzel defines a computer program simply as:

> "... a sequence of instructions that tells the computer a step at a time which operations to perform."[3]
> [My emphasis]

In a paper entitled "Computer Programs - Art or Science",[4] Perry and Killgren define 'an instruction' and a 'computer program', in greater detail, as follows:

> "An instruction ... specifies data to be operated on and the operation to be performed. A computer program is a set of instructions designed to cause the computer to execute some task. In other words a computer program is the means whereby a computer is caused to perform some function."[5]
> [My emphasis]

The most discussed, and arguably the most widely accepted, definition is contained in the World Intellectual Property Organisation's Model Provisions on the Protection of Computer Software.[6] In Section 1(i) therein, a computer program is defined as:

> "... a set of instructions capable, when incorporated in a machine-readable medium, of causing a machine having information-processing capabilities to indicate, perform or achieve a particular function, task or result."[7]
> [My emphasis]

This definition was included in a recent bill laid before the US House of Representatives.[8] The bill was drafted with the intention of clarifying US copyright laws as they applied to computer software.

While each of these definitions is subtly different from one another, in all of them a computer program includes instructions designed to cause a machine to operate in a certain way. The term 'computer program' or simply 'program', as adopted hereafter, comprises merely these two essential features, namely a set, or sequence, of instructions designed to cause a computer to perform some task. This definition closely follows the above WIPO definition.

While there is no agreed definition of a program, it is generally agreed that programs can be divided into two groups, 'applications' programs and 'systems' or 'control' programs.[9] However, the distinction between them can be arbitrary;[10] it sometimes depends on the nature of the computer application. One man's control program is another's application program. In general terms, application programs are 'problem oriented' or as Frenzel describes:

> "An applications program ... is the sequence of instructions designed for a specific situation. Applications

> programs cause the computer to perform useful services."[11]

Examples include payroll, stock control, purchase ledger and games programs. In each case the program is designed to do a specific task. Some applications programs can be used, without modification, by a large number of computer users, each having the same, or substantially similar hardware. Others are tailor-made to a computer user's particular requirements.

On the other hand, systems or control programs are 'computer oriented', or as Frenzel describes:

> "Systems programs make the computer easier to use. They simplify and speed up the programming process. They make computer operation and programming faster and more efficient."[12]

The distinction between applications and systems (or control) programs has been described by one programmer as analogous to prose and grammar.[13] Applications programs are to prose what systems programs are to vocabulary and punctuation. In operation an applications program uses the systems programs to execute its instructions.

Systems programs also have other, 'housekeeping' functions. In particular, certain systems programs are designed to facilitate efficient data transfer between elements of the computer system (e.g. from secondary back-up storage to the computer's primary memory). In addition, they aid program writing, co-ordinate and schedule operations within the computer, control

input and output of data and monitor the execution of the applications program.[14]

A second distinction between the two types of programs is that systems, or control, programs are designed to optimally utilise the resources of the computer, whereas applications programs are tailored more to the perceived needs of the computer user.

Within the two categories of program, further divisions can be made. In his book "Operating System Elements: a user perspective", Calingaert classified systems programs generally as 'language system', 'operating system' and 'utility system' programs.[15] 'Language system' programs refer to those designed to translate other programs from one computer language to another, sometimes from languages designed to be easily comprehensible to human beings, to languages oriented to machine communication. The 'meaning' of the instructions comprising the original program is preserved in translation, but their expression is likely to be completely different.[16]

Calingaert uses the term 'operating system' programs to refer to the collection of programs designed to control the resources of the computer system for a network of users.[17] Frenzel uses the same term to mean those programs designed to supervise and sequence computer operations for a single user.[18] The protection of applications programs used in a network of computer users presents a host of additional problems to the program supplier, compared with the problems inherent in protecting

programs supplied to customers each having individual, isolated computers. In particular, it is clearly desirable for the supplier to be able to restrict access to his program by some means (legal and/or technical), to authorised computer users only. Those computer users in the network who have not bought, or licensed, the program should not be able to obtain access to it. These additional problems are discussed elsewhere,[19] and are not considered in detail here. Rather this, and subsequent chapters of this thesis, concentrate on the problems of protecting computer programs designed for a community of end users, each having an individual, isolated computer. For this reason, Frenzel's interpretation of the term is adopted hereafter.

Calingaert's third category, 'utility programs' refer to those designed to transfer programs from one medium to another.[20] Computer programs supplied to users are not expressed in a form in which they can be directly implemented. Rather each program first has to be 'loaded' into the computer, prior to execution, that is, it has to be translated from one medium of expression suitable for transporting it to the end user, to another medium suitable for execution of the instructions therein by the computer system. For example, many computer programs are supplied to end users on magnetic disks. In order for the program to be run, it must first be transferred to the Random Access Memory, or RAM, of the computer. In one embodiment of RAM, the program is expressed as electric signals controlling the conduction of an array of transistors etched on a silicon substrate.

Computer Languages

All computer programs are expressed in one or more computer languages. Computer languages, or programming languages, are analogous to traditional human languages in that each has its own collection of symbols and syntax.[21] But, unlike human languages each is designed to communicate with a machine, not a human being.[22]

The expression of a program which drives the computer's integrated circuits is called 'machine code', the program being expressed, for digital computers, in a binary machine language. This language comprises two symbols, '0' and 'I', termed binary digits or 'bits'. These can, for example, be represented by the conduction/non-conduction states of a transistor, or the opposite polarities of a small portion of magnetic media. Instructions comprising the program are each represented as a string of bits.

However, writing a program in machine language is extremely laborious because the language is so primitive. Typographical errors are easily made and are very difficult to detect subsequently. Thus developments in programming can be characterised by innovations designed to make program writing, 'coding', easier.[23]

The first such development was 'assembly language'. Simple mnemonics replaced the strings of binary numbers representing the instructions comprising the program.[24] An 'assembler' program would also check each line of the assembly code program for syntax errors, and provided it was bug-free,

translate it into its machine language equivalent for execution by the computer. However, writing programs in assembly language also had its limitations, particularly in long programs comprising many thousands of instructions. The previous disadvantages re-appeared, namely that errors were easily made and difficult to detect subsequently.

The development of 'high level languages' made program writing easier. Unlike assembly language programming, there no longer exists a one-to-one correspondence between the 'vocabulary' of the high level language and its machine language equivalent. Rather each high level language instruction is translated into a block of machine-readable instructions by a translator program (or programs) in the computer, prior to execution. Therefore the disadvantages inherent in writing long assembly language programs are alleviated, for effectively the programmer writes an abbreviated version of the instructions subsequently executed. Furthermore, the development of high level languages implies that programmers no longer need an in-depth knowledge of the computer hardware and associated operating systems software. They merely require expertise in the relevant high level language in order to write programs.[25]

High level languages are each designed for particular types of problem or application.[26] For example, FORTRAN is designed for mathematical and scientific applications; COBOL is designed for business applications. Furthermore, some high level languages, for example BASIC - Basic All-purpose Symbolic Instruction

Code,[27] closely resemble certain English statements.

There are now 'program generators' in which a keyboard operator types in English phrases, or even sentences, which are then each translated into a computer program or routine. Thus the operator no longer needs to have expertise in a high-level language in order to write programs.

Execution of Programs in Computers

The high level language representation of a computer program is called 'source code'. Source code cannot directly control the integrated circuits of computer hardware. Rather, the program has to be translated into a machine-readable form prior to execution. Language system programs, referred to above, are designed to translate instructions of the program from a source code form to an 'object code' form, a machine-readable translation of the program. Depending on the high level language used, source code can be 'compiled' or 'interpreted' into object code. The difference between compilation and interpretation of source code lies in the mode of operation of the respective compiler and interpreter language system programs.

A compiler performs several passes over the source code. In addition to translating the program into object code, it also identifies and executes several instructions of the program. For example, any 'jump' instructions are implemented.[28] Thus if the same source code instruction is referred to at several points

in the program, the compiled version contains the corresponding object code instructions at each of these points. The compiler also performs various housekeeping functions. One advantage of compilation lies in the speed of data processing. The instructions of the compiled program are arranged in the order in which they will be executed. In operation, the computer merely steps from one object code instruction to the next, sequentially. A second advantage is that once compiled, the compiler is not required in the execution of the object code program. The compiler can therefore be removed from the computer's memory, releasing that part of memory for other uses.[29] However, a disadvantage of compilation lies in the inefficiencies of translation. Large portions of memory are sometimes required to store the same routine if it is repeatedly used in processing.

Alternatively, an interpreter program translates each line of source code into object code, which is then executed, before it translates the next line of source code. Unlike a compiler, an interpreter does not pass over the source code several times. Rather, translation and execution of the program are performed in one pass.[30] The primary advantage of an interpreter over the compiler is that it produces faster results, for it does not translate every line of source code into object code prior to execution. For this reason, interpretation of programs provides a most valuable aid to program writing, for typographical errors in a line of source code are detected almost as soon as they are written.[31] One of the disadvantages, however, is that the interpreter must

reside in memory, because translation and execution of each instruction immediately follow one another. Therefore there exists a memory overhead, limiting the length of the source code program that can be stored in memory.[32]

The 'instruction set' is the collection of basic logic and arithmetic functions of the computer system. These functions are used to execute the object code instructions of the program. Instruction set functions are implemented either by dedicated integrated circuits or a combination of less complex circuits and associated 'micro code' or microprograms.[33] In the former, the object code instructions are executed in hardware directly. In the latter the object code instructions drive a program, or programs, stored in Read-Only-Memory (ROM) which in turn drive the computer's circuits.[34]

Development of a Computer Program

The development, or evolution, of a computer program can be characterised as comprising six steps or stages.

1. The problem to be solved by the program is defined.[35] Alternatively, the idea, concept or notion underlying the program, as the case may be, is conceived.[36]

2. A solution to the problem 'the algorithm' is developed and written down.[37] (Alternatively, the idea, concept or notion is written down.) Frenzel defined the term 'algorithm' as:

> "... a set of rules, processes or steps that define a solution to a specific problem."[38]

Algorithms can be represented in many different embodiments including a written description of each of the steps required to solve the problem and/or flowchart representing these steps and interconnections between them.

3. The algorithm is expressed as a source code program. Each step is expressed as one, or more, high level language instructions.[39] Also, explanatory comments are included in the source code to guide other programmers through the program.

4. The program is transferred to the computer.[40] It is now analogous to a prototype.

5. The program is 'run' and tested. This 'benchmarking' stage transforms the program from a prototype to a reliable, proven, and therefore valuable commercial product. Errors, or 'bugs', typographical and logical, are discovered and corrected.[41] The correction of errors may entail returning to one or more of the previous stages.

6. Support documentation required to use the program effectively, is written. Such documentation includes: (a) a general description of the problem; (b) a flowchart outlining the algorithm; (c) a copy of the source code; (d) a technical specification of the program, including details of hardware and operating systems software required to implement it; and (e) a manual for the end user, specifying step-by-step how to use the program effectively.[42]

Program development has been described in terms of several

successive discrete stages. In practice, all may be combined into one continuous process of program evolution. Many programs are developed interactively, that is with the aid of a computer, keyboard and display, in which the coding, debugging and documentation stages proceed in parallel.[43]

Also, developing a program is a highly creative activity. There are many different ways of solving a given problem. Indeed, programmers have recognisable coding styles analogous to writing styles of authors.[44] Furthermore, it is a slow, labour intensive activity, requiring highly skilled manpower. A computer program is, more often than not, the result of a large investment of time, money and expertise. Thus each stage in the development of a program represents an investment. From the point of view of the software supplier, it may not necessarily only be in respect of his final product, the proven computer program, that he requires protection. In particular, an original algorithm, developed in stages (1) and (2) above, can be the program's most valuable asset, and therefore that which the supplier would be most keen to protect.[45] Thus, while this thesis is primarily concerned with the protection of computer programs, reference will be made, where necessary, to the protection of algorithms.

Computer Software

In common with computer programs, there is no universally adopted definition of 'computer software'. However, it is

generally agreed that it includes information other than the program *per se*. In the WIPO Model Provisions, software is defined as comprising the 'computer program', 'program description' and 'support material'.[46]

The term 'program description' is defined therein as:

> "... a *complete* procedural presentation in verbal, schematic or other form, in sufficient detail to determine a set of instructions constituting a corresponding computer program."[47]
> [Emphasis is mine]

The definition requires that the material be a *complete* description of the program to be considered as a 'program description'. Thus an algorithm expressed as a flowchart is included in the definition, for a corresponding computer program can be derived from it, but a general description of the program is excluded from it, for additional data would be required before a program could be determined.

The term 'support material' is defined as:

> "... any material, other than a computer program or a program description, created for aiding the understanding or application of a computer program, for example problem descriptions and user instructions."[48]

The definition clearly includes a general description, user manual and other material useful in understanding or applying the program.

Frenzel includes the items specified in the WIPO Model Provisions as well as:

> "... all additional procedures and documentation associated with the programming process."[49]

This includes post-sale maintenance and updating services provided by many software firms to their end users.

For the purposes of this thesis, the term 'computer software' or simply 'software', as used hereafter, follows the WIPO Model Provisions. Therefore, software comprises the program (in any embodiment), and all relevant material.

COMPUTING SERVICES INDUSTRY

Many of the problems of protecting computer programs arise from the new widespread use of computers in commerce, industry and the home. The low cost of microcomputer hardware, together with the realisation that practically any control and/or data processing task, in any physical environment, can be performed by a suitably programmed microcomputer, have led to rapidly expanding computer hardware and software markets. According to a recent survey of UK microcomputer retailers, sales of microcomputer hardware have trebled from June 1982 - March 1983.[50] Its findings revealed that up to March 1983, 1.35 million microcomputers were installed in the UK, one million of which were sold in the previous 12 months.[51] The large number of microcomputer users this 1.35m. figure represents, provides a large market for compatible software products. Therefore, in conjunction with the growth of a microcomputer hardware industry, the computing services industry has grown in recent years.

This section briefly describes some of the technological and commercial developments that have influenced the development of the computing services industry, not only in the UK but also overseas. It is not intended to be a history of the industry, rather provide a context within which to place the problems of protecting computer programs from the point of view of the supplier.

Three developments have had a pervasive influence on the industry:

(a) IBM's 'unbundling' decision in 1969;

(b) the application of microelectronics in computing; and

(c) the development of Control Program for Microcomputers (CP/M).*

(a) IBM's 'unbundling' decision and its consequences

From the birth of the computing industry until the late 1960's, programs were supplied to end users mainly by computer hardware manufacturers. Programs formed part of a hardware/software package including the computer hardware, and post-sale maintenance services in addition to software.[52] The computing services industry then comprised a small number of companies supplying certain programs, or providing certain services, that were not cost effective for the computer hardware manufacturers themselves to provide.[53] In particular, software companies provided three services:

---

* 'CP/M' is a trademark owned by Digital Research Inc.

(1) they advised computer users. Furthermore, they modified standard programs, tailoring them to the individual needs of the user;

(2) they developed specialist programs for particular computer users as an extension to the services outlined in (1) above; and

(3) they wrote programs, under contract, for computer hardware manufacturers.[54]

Computer hardware manufacturers regarded program development costs as an overhead compared with hardware development costs.[55] Their customers paid for the hardware (the electronic circuits comprising the computer), with programs being given away free, or nominal charges being made in respect of them. However, improved methods in the manufacture of integrated circuits decreased the hardware price component in the package, so much so that program development costs could no longer be considered merely as an overhead.

Thus, in 1969, IBM began to charge its customers separately for its hardware and software, a development which has become known as 'IBM's unbundling' decision. Programs were no longer supplied free or at nominal prices. Rather, the prices more closely reflected the investment sunk into creating and supporting them.[56] The decision was also due, in part, to an impending anti-trust law suit in the US in respect of IBM's software policies.[57]

The other computer hardware manufacturers (including the UK's only major computer hardware manufacturer, ICL) quickly followed IBM's lead.[58]

The IBM decision had the effect of promoting competition

within the industry. The computing services industry rapidly expanded in the 1970's. By the end of the decade both the UK and US industries comprised a large number of small software firms. In the US, the Association of Data Processing and Services Organisations (ADAPSO) had 46 members in 1973,[59] 308 in 1978[60] and now has a corporate membership of 500.*[61] In the UK, the Computing Services Association (CSA) had 93 members in 1975 and now has over 180 members.[63] The Computer Retailers Association (CRA) set up in 1979, now has 47 members.[64]

The services provided by the companies in the computing services industry have increased considerably in the 1970's and early 1980's.# Originally, the industry comprised computer bureaux, hardware manufacturers and a small number of independent computer program suppliers. The industry now includes these, as well as new types of company specialising in the marketing of programs, and particularly software publishers, distributors, and retailers. Many firms provide a combination of these

---

* According to one estimate, the number of software firms now in existence in the US is 6000, whereas 10 years ago the 1973 ADAPSO figure of 46 comprised the majority of companies in the industry.[62]

\# According to a recent study of the UK Computing Services Industry,[65] the returns of 140 companies in the financial year 1980/81 totalled £612m., representing an annual growth rate of 14% for the industry as a whole. However, if the figures are corrected to 1975 prices, it indicates a much smaller growth rate of 4%. The report argued that this apparent anomaly arose from the difficulty in interpreting figures, for a 'heterogeneous' industry. It reported that some parts of the industry were rapidly expanding (e.g., software houses) whereas others were static or even contracting (e.g., computer bureaux).[66]

services. The findings of a recent survey of 156 mainly UK software companies revealed that while 104 wrote one, or more, types of application programs, 48 of them also specified one, or more, of the following related activities: software publishing, retailing and distribution, hardware manufacture and systems programming.*

It is evident that the industry is rapidly expanding, but at the same time, rapidly changing. The traditional data processing services provided by centralised computer bureaux has given way to data processing at the end user's premises, following the introduction into the market of cheap microcomputers. This, in turn, has given rise to companies specialising in the marketing of compatible software products to these computer users.

Microelectronics in Computing

All digital computers comprise four basic sections, memory, arithmetic logic unit, input/output unit and control section.[67] It is this combination that distinguishes the digital computer from other data processing machines, for it is capable of making decisions and changing its method of data processing, from the information it is working with, according to its program. In common with mainframe and minicomputers, a 'microcomputer' has each of the four basic sections of a digital computer. While it is sometimes difficult to distinguish between certain

---

* The results of this survey are discussed in greater detail in chapter 4 below.

certain minicomputers and large microcomputers, all microcomputers contain an integrated circuit called a 'microprocessor'.[68]

'The microprocessor comprises a programmable single large-scale integrated circuit, typically 5 millimetres square, designed to perform the combined functions of an arithmetic logic unit and control section in a digital computer, to form a central processing unit (or CPU).[69]

The development of the microprocessor stems from the application of innovations in the manufacture of semiconductor electronic circuits to the fabrication of circuits for computers. These innovations tended to increase both the reliability of electronic circuits and the number of circuit components (transistors, resistors and capacitors) that could be etched onto a silicon substrate, while at the same time decreasing the size of such circuits. A detailed account of the history of semiconductor electronics is contained in "Revolution in Miniature: the history and impact of semiconductor electronics" by Ernest Braun and Stuart Macdonald.[70]

The value of the microcomputer lies in its price, reliability, size and versatility. A supply voltage, clock (usually an oscillating crystal), keyboard, display and a program are all that are required to 'run' the microcomputer.[71] Some microcomputers retail for hundreds and even tens of pounds, and therefore there now exists a very large market in the UK and elsewhere for home and business microcomputers, and associated software.

Control Program for Microcomputers (CP/M)

CP/M was one of the first operating systems software products designed to run on certain 8-bit microcomputers. It has become one of the most widely used operating systems of its kind.[72] According to one estimate there are over one million authorised end users of CP/M worldwide.[73] Since its introduction into the market in the mid-1970's, CP/M has become a standard feature in many microcomputer systems.

CP/M evolved from research into programming languages designed for the INTEL 8008 microprocessor. In 1973-74, Dr. Gary Kildall developed a high level programming language called 'PL/M' to aid the writing of programs designed to run on the 8008. In devising PL/M, he discovered that program writing could be made far easier if the programmer used a combination of a terminal, microcomputer and diskette drive. However, it lacked an 'executive' or operating systems program required to co-ordinate and control the operation of these elements in a computer system. Thus, as an offshoot to PL/M, Dr. Kildall conceived and developed the software now known as CP/M.[74]

CP/M is not one program but a collection of 26 interconnected programs. These can be divided into two groups, the Basic Input and Output System (BIOS) and the Basic Disk Operating System (BDOS). The essence of CP/M is that the BDOS is substantially machine independent, whereas the BIOS is machine dependent and has to be modified depending upon the particular hardware configuration of the computer system.[75]

It was originally designed to run on microcomputers using 8-bit Intel 8080 or Zilog Z80 microprocessors. Although there were other 8 bit microprocessors on the market at that time (e.g., MOSTEK 6502), these two together accounted for a large portion of the microprocessors then being used. CP/M became a de facto standard operating system for these two microprocessors. There are two reasons accounting for the popularity of CP/M. First, it made program writing easier, and secondly, it promised large markets for the resulting software. CP/M now has one of the largest libraries of compatible software products. According to one estimate there are between 6000-8000 currently available products in the US requiring CP/M for their implementation.[76]

THE PROGRAM PROTECTION PROBLEM

The device most commonly used to protect computer programs is the contract or non-exclusive licence. Computer users do not buy a copy of the program outright, rather they buy a limited right to use the copy for a specified term. While the conditions of 'sale' vary widely from one software supplier to another, the following (or variations thereof) are usually included:

(a) that the software supplier (licensor) retains copyright and all other intellectual property rights in the programs and associated documentation;

(b) that the customer (licensee) is prohibited from copying the documentation and disk, or other media, containing the programs except for the purposes of back-up;

(c) that the licensee is permitted use of the programs on an authorised (named) computer only;

(d) that the licensee is bound by an obligation of confidentiality to the licensor, under which he is not allowed to divulge details of the software to third parties without the licensor's prior consent; and

(e) that once the licence has expired and not renewed, the licensee undertakes either to destroy all copies of the program and associated documentation, or return them to the licensor. In both cases, the licensee may be required to make a written declaration specifying compliance with one or other of these conditions.[77]

It seems to be generally accepted, at least among CSA members, that the non-exclusive licence provides effective and adequate protection to software firms supplying so-called 'mainframe' and 'minicomputer' programs.[78] However, it is debatable whether the effectiveness of this protection stems primarily from a responsible computer user community, or whether the protection is inherent in the programs themselves. There are a number of features of such programs, and the computers on which they are designed to run, having the effect of enhancing the protection provided by the licence. These include:

(1) <u>the 'know-how' associated with a program</u>. The development of a program generates information, tricks of the trade, or programming techniques (perhaps even the rejection of obvious techniques for better, less obvious, ones), information whose

importance may not necessarily be obvious from the resulting software. This know-how is required in the subsequent development of modifications, or enhancements, of the original program.[79] Since such information is usually regarded as company confidential, an unauthorised supplier would be unable to effectively support customers supplied with the illegal reproductions.

(2) The regular post-sale maintenance and services provided by many software suppliers.[80] For certain computer programs, their utility for the computer user depends upon their being up-to-date. For example, a payroll program includes a National Insurance subroutine which would probably require modification following changes announced in the Budget. Unless the computer user had an 'in depth' knowledge of the structure of the program, he would be unable, without help from the supplier, to modify the program himself. Effectively, customers are dependent upon the supplier for continued use of programs whose parameters must be regularly updated, because they (the parameters) represent changing external conditions. Therefore, the advantages gained by the unauthorised program user is likely to be shortlived, for sooner or later he would have to approach the supplier in order to obtain crucial modifications or updates of the software. Once approached, he is likely to be found out, for the actual circumstances of his purchase would then be discovered.

(3) The supply of programs in object code from only. To aid the maintenance of programs, they are usually supplied to

customers in object code form only. The programs are not expressed in a form understandable to the majority of computer users (nor indeed programmers). It is unlikely therefore that computer users could support such programs themselves, even if they knew the proposed enhancements of the supplier, because they would not know what to modify in the object code.[81]

(4) The market for programs. Some programs are written for one, or a small number of computer users. Little advantage would be gained from copying without authorisation for there exists, at best, a small market for them.[82]

(5) Small computer user community. The various mainframe and minicomputer user communities tend to be quite small, so that for a particular program there is a strictly limited number of compatible computers and operating systems programs.[83] Therefore, in common with (4) above, there is, at best, a small market for unauthorised reproductions of the program.

Clearly the scope of protection provided by each of these features depends upon the particular program. However, none prevents the licensee from deliberately breaking the conditions of his licence, rather each removes some incentive for him to do so.

Some of these features exist in certain microcomputer programs, and therefore, in common with 'mainframe' and 'minicomputer' programs, they provide some limited measure of inherent protection to the software supplier. However, others do not, and particularly features (4) and (5) above. Unlike

mainframe and minicomputer programs, there exist vast markets for standard microcomputer applications and systems programs.[84] Opportunities exist for unscrupulous end users and dealers to copy popular microcomputer programs, and sell the reproductions to unsuspecting computer users of the appropriate microcomputer user community.

Supplying Software to a Mass Market

In practice, a supplier's licence and intellectual property rights become his only means of protection. However, it is debatable whether the licence is an appropriate form of protection for microcomputer programs designed to cater for a mass market.[85]

Many software houses do not have the necessary retail organisation or expertise to market their programs effectively to a mass market. As a result they usually market their programs through a network of intermediate dealers. While practice varies from one software house to another, many do not permit their dealers the right to copy their programs. Many software firms supply their dealers with packaged software, each package comprising a sleeve containing a copy of the program on magnetic disk, a registration card and also perhaps a user manual. In theory, when the dealer makes a sale, he passes the package unopened to his customer. He (the customer) returns the registration card to the software house. The dealer informs the software house, or supplier, of his sale and remits the

royalty to him. The supplier should then be in a position to check that he is receiving the correct revenue, for each sale is verified by the return of the registration card from each of the dealer's customers.[86] However, this procedure is not foolproof, for some customers fail to return their cards, and some dealers fail to inform their suppliers of their sales.

Some dealers open the package, copy the program and sell the reproductions to end users without registration cards. If the dealer fails to inform the supplier, he (the supplier) does not have a record of the transaction. Thus, he has no means of checking the amount rightfully owing to him. Likewise, some end users could, without the authorisation of the software house or the dealer, copy the program and sell reproductions on their own account.

In each case the risk of being found out is small, particularly if the number of end users and dealers is so large that it is simply not feasible, nor cost effective, for the software house to conduct an investigation into the source of such reproductions. The supplier may suspect that his licenses or agreements have been breached, but he does not know by whom. Indeed in many instances a software house discovers infringements by chance.[87] Furthermore, even if the infringer is identified, the costs incurred in attempting to enforce copyright in his software, deters all but the most determined, and rich, of suppliers from seeking redress in the courts.[88]

Thus, the software supplier is, to a large extent,

dependent upon the honesty of the software dealer and end user for remittance of the revenues to which he is entitled.

Abuse of the Supplier's Rights

While a large proportion of end users and dealers are honest, unauthorised copying and so-called 'software piracy' of some programs is endemic in the US and UK.[89] Games and cheap personal and business software are usually cited as those programs regularly being copied without authorisation.

According to a lawyer in one US-based microcomputer software firm supplying a popular wordprocessing software product, it receives 20% of the royalties to which it is entitled from its US market (i.e., he estimates that 4 out of 5 copies in the US are unauthorised).[90] Furthermore, he estimates that the company receives only 10% of the royalties, to which it is entitled, from its European markets.[91]

An article in the weekly UK trade journal "Computing" recently reported that:

> "... the UK software industry loses millions of pounds a year through piracy."[92]

Indeed, one software house offers a reward for information on the use of illegal copies of one of its software products.[93]

There are clearly many serious instances of unauthorised copying of programs, but the extent to which it goes on is largely unknown, for copying usually occurs in private. For example,

within one company, supplying a program which is reputed to be widely copied and distributed without prior authorisation, estimates on the degree of illegal copying vary from 25% (that is, 1 in every 4 copies are unauthorised) to 500% (that is, only 1 in 5 copies are authorised).[94]

While there is no agreement on the extent of the problem, it is generally agreed that there are two main types of unauthorised copier, those who copy privately and those who copy for profit.[95]

Private Copying

Private copying is where an authorised end user copies the program and supplies it to another end user in breach of his licence. Although many unauthorised reproductions are usually supplied free of charge, each one constitutes a lost sale for the supplier. Private copying is considered by many to constitute the greater loss of revenue, and also the more difficult to stamp out.[96]

The existence of private copying has been attributed to a number of factors including:

(a) that software is overpriced, thereby encouraging copying, or swapping, of expensive programs;[97]

(b) that end users are not aware that copying is harmful to the livelihood of their suppliers;[98] and

(c) copying is considered a 'legal', rather than a 'moral' crime.[99]

The extent of the loss of software revenue depends upon both the price of the program and the number of end users

supplied from the unauthorised source. Many instances of private copying are from one person to another, or from one person to a small circle of contacts around him/her. In each case, while the software firm would be keen to keep this unauthorised group of end users from expanding, the investment of time and money expended in chasing them up, usually outweigh the revenue lost. There are rarely, if ever, cases reported concerning a software firm prosecuting an end user for alleged copyright infringement or breach of contract. Rather, grudgingly perhaps, some software firms bear the loss of revenue, and merely warn the user. However, some end users each have a large circle of contacts, and therefore it does not always follow that lost software revenue can readily be written off. Private copying within large corporations, user groups and educational institutions are regarded by many software firms as constituting serious losses of revenue, because in each case the circle of contacts is so much larger.

Software Piracy - Copying for Profit

Copying for profit is where an end user, or intermediate dealer, reproduces the program and sells it to unsuspecting end users without informing the authorised software supplier of these sales. The 'pirate' pockets the royalties rightfully owing to the software supplier from these transactions. Although the number of unauthorised copies made by software pirates is considered, by many, to be less than those made by

private copiers, the loss of revenue from piracy is regarded as being the more serious. In this respect, piracy by software dealers was repeatedly mentioned in interviews with software executives, because in each case, the circle of contacts is so much larger compared with the contacts of a computer user. Many software suppliers are inclined to sue dealers found to be in persistent breach of contract. In a number of well publicised law suits, some software suppliers in the US have successfully prosecuted pirates for copyright infringement of their programs.[100]

CONCLUSIONS

The threat of widespread unauthorised copying and piracy, together with the uncertain scope of intellectual property rights for software, have led many software firms to seek alternative methods of protecting their vulnerable mass-market programs. These companies have invested in and developed technological forms of protection, written into their programs. These technological measures are designed to prevent, or deter, those activities that effective intellectual property rights would, in most instances, deter.

CHAPTER 2

C O P Y R I G H T

INTRODUCTION

From the point of view of the software supplier, the major disadvantage of the non-exclusive licence or contract is that it is binding only on the licensee.[1] Should the licensee supply a third party a copy of the program, in contravention of his licence, the licensor (the software supplier) cannot sue the third party for breach of contract. He may, of course, be able to sue the licensee. However, it may be the third party that the original software supplier is most keen to stop, particularly if he is selling a large number of illegal reproductions of his program. The original supplier's main recourse to law against a third party rests upon the intellectual property rights he enjoys in his software.

Many legal experts, in the UK and overseas, regard copyright as being the most appropriate form of intellectual property currently available for the protection of computer software.[2] The principal attraction of copyright in the UK is that it subsists for 50 years from the moment the work is created. There is no formalities to be adhered to prior to obtaining the right. Furthermore, through the Berne Union and Universal Copyright Convention, UK copyright owners can obtain copyright protection overseas, under the same conditions as nationals of the signatory

countries.

One argument used to justify the applicability of copyright law for the protection of programs is that they are often embodied in human-readable form, and therefore that they are similar to other types of works that have long been regarded as suitable subject matter for copyright protection.[3] However, since programs are ultimately designed to condition, or control, electronic circuits in computer hardware, and therefore that they are also commonly expressed in machine-readable embodiments, it is open to question whether this argument is wholly appropriate.

In the UK copyright has statutory foundation in the Copyright Act 1956.[4] This Act was drafted when computer programming had not yet been recognised as an art including works capable of copyright protection, and therefore nowhere in the statute is there mentioned 'computer program', 'computer', 'software', or 'algorithm'. Furthermore, case law in the UK has not yet established the subsistence of copyright protection for computer programs. But there have been a number of cases heard overseas that could have a bearing on the scope of UK copyright for programs. Thus, to determine the scope of protection copyright can provide, this chapter will consider relevant subsections of the Copyright Act 1956 in detail, together with reported case law.

In addition, the copyright laws of the US will be briefly considered. There is a large, and growing, body of US case law clarifying both the embodiments of programs considered suitable

for copyright to subsist, and the scope of the copyright owner's rights. While such cases have no precedential value in the UK, it is probable that these developments will have an influence (albeit a passive influence) on the evolution of copyright law in the UK.[5]

COPYRIGHT ACT 1956 - Problems in Interpretation

The 'works' protected by the 1956 Copyright Act are classified into classes. These classes are divided into two parts. Part 1 concerns original literary, dramatic, musical and artistic works and part 2 concerns sound recordings, cinematograph films, television and sound broadcasts. In general, the conditions for the subsistence of copyright, and the rights of the copyright owner are specified for each class of work.

For computer programs and software in general, the reference to sound recordings and the like exclude part 2 classes from consideration.

Referring to part 1 classes, section 2(1) of the statute states:

> "Copyright shall subsist, subject to the provisions of this Act, in every original literary, dramatic or musical work ..."[6]

Programs are clearly not included in the classes of dramatic and musical works. Therefore the class of copyright works which could include programs is 'literary works'. While the scope of the term 'original literary work' as used above, has, through case law, resulted in its having a wider interpretation than that

commonly used, it is uncertain whether it extends to computer programs. This is primarily because there have been only a small handful of reported cases concerning the subsistence of copyright in programs.[7] Furthermore, all of these have been heard at the interlocutory injunction stage of litigation. No program copyright case has yet gone to full trial; and also no such case has been argued in the Court of Appeal or the House of Lords. Therefore, there are currently no precedents having the effect of establishing the suitability of programs for inclusion in one or more classes of protectable works.

In spite of this however, it is now thought that some forms of expression of programs (and software documentation in general) are 'original literary works', and therefore attract copyright.*[9] Furthermore, the expression of algorithms in flowcharts is also believed to be included within the scope of the term 'artistic work' as used in the Act. In addition, evidence for the subsistence of copyright in programs is provided by a number of out-of-court settlements.[10]

However, the applicability of copyright depends upon adopting expansive interpretations to certain terms used in the Act. Interpretation of the statute is further complicated, for

---

* In Northern Office Microcomputers (Pty) Ltd. v. Rosenstein (1982) FSR 124, a South African court held that the source code version of a computer program was included in the 'literary works' class of works protected by the South African Copyright Act. Since the wording of the Act is identical in this respect to the Copyright Act 1956, it is probable that, in a similar case in the UK, the High Court would follow this precendent.[8]

the meaning of terms change from one section of the Act to another. As a result, the Act has been described, in the report of the Committee to consider the law on copyright and designs (termed 'Whitford Report' hereafter), as:

> "... a remarkable feat of draftsmanship but, even if it is a draftman's dream, it has proved to be a nightmare to those who have to try to understand it whether as laymen for their own purposes or as lawyers seeking to guide their clients."[11]

A major problem lies in the inadequate definition of 'literary work' in section 48, the interpretation section, of the statute. In the Act, the term is used in two different contexts: (a) in the requirements for the subsistence of copyright; and (b) in specifying the rights of the literary-copyright owner. For (a), the term is interpreted as the book, manual, article, or program as the case may be, the product of the author's literary efforts. For (b) however, it is interpreted as the skill and/or labour expended by the author in the creation of his work. It follows that from (a), literary copyright requires fixation of the author's skill and/or labour in an acceptable material embodiment(s)[12]; but from (b), the 'literary work' actually protected is not only the work rather, to a limited extent, it includes the intangible skill and/or labour expended in producing it.

Thus for computer programs many uncertainties remain, and particularly (1) whether or not machine-readable embodiments of programs are 'literary works' and (2) the extent of the

copyright owner's rights in his program.

An Interpretation of 'Original Literary Work'

The term 'literary work' is only partially defined in Section 48 as: "including any written table or compilation".[13] The term 'writing', referred to therein, is likewise only partially defined as including:

> "... any form of notation, whether by hand or by printing, typewriting or any similar process."[14]

Since the definitions are inclusive, it has been left to the courts to determine the scope of the term 'original literary work' as a class of protected copyright works.

It has long been established that the class includes works lacking in any aesthetic merit.[15] It has also long been accepted that works written in languages such as braille, are literary works protected by the Act.[16] Furthermore, in Anderson v. Lieber Code (1917) 2 K.B. 469, a telegraph code designed to minimise errors in the transmission of messages by morse code was deemed a copyrightable literary work, even though the works comprising the code were in themselves meaningless. Thus there is no requirement that a work has to be seen and/or directly understood by a human reader in order to be protected by literary copyright.[17] Therefore from the tenor of these decisions, a source code listing of a computer program printed on paper is probably included in the class of protected literary works, as

are other items of software expressed on paper, e.g., user manuals and technical specifications.

However, it is uncertain just how far this interpretation extends. In particular it is uncertain whether: (1) programs expressed as an array of electronic circuits etched onto an integrated circuit 'chip'; (2) programs contained in a Read-Only-Memory (ROM), Random-Access-Memory (RAM), or Programmable-Read-Only-Memory (PROM); and (3) programs expressed as an array of polarised magnetic poles contained in a disk coated in a magnetic material, are included in the class of protected literary works. Furthermore, this is not now merely a matter of academic interest, for while many programs are first embodied on paper, and therefore probably attract copyright, it is becoming more and more prevalent that programs are created with the aid of a word-processor. Such programs are displayed on a visual display unit (VDU) and then, if required later, stored in the computer's RAM. The proven and therefore economically valuable computer program may never be expressed in conventional written form as a printed source code listing. It is conceivable that programs expressed in certain machine-readable embodiments (such as the ones above) may <u>not</u> be considered 'literary works' and therefore may not be subject matter suitable for copyright protection.

Section 49 of the Copyright Act 1956 contains supplementary provisions on interpretation. Subsection 4 therein appears to extend the interpretation of literary works to these machine-readable embodiments for it states:

> "... a literary ... work was made [when] it was first reduced to writing or some other material form."[18]

But statute and/or case law has yet to establish whether or not ROM, RAM and magnetic disk embodiments of a computer program constitute 'writing or some other material form' as used above.

Following the Whitford report, in July 1981, the Government presented to Parliament a Green Paper on copyright law reform.[19] Chapter 8 of this document outlined the problems of copyright protection for computer programs, and furthermore made a number of recommendations. In particular, the Government proposed to make it explicit in new legislation that:

> "... computer programs attract protection under the same conditions as literary works."[20]

The proposal implies that programs possessing originality should obtain copyright in the same way as it now subsists in 'literary works' as interpreted in the Act. Furthermore, the Government considers that embodiments of programs suitable for copyright to subsist should extend to their fixed machine-readable forms, for it proposes that:

> "... copyright protection should extend to any form from which they can be reproduced."[21]

However, the enactment of a new copyright act containing these recommendations is not expected in the foreseeable future. For the moment therefore, the suitability of machine-readable embodiments of programs, as literary works protected by the Act, remains

uncertain.

Copyright in a literary work subsists only if the work is deemed 'original'.[22] The necessary degree of originality is not high, for, in the Whitford Report, it is stated:

> "The claim to originality means no more than that the creator of the work can truthfully say 'this is all my own work'".[23]

A work is not considered original if it is merely a slavish copy of an earlier work.[24] However, this does not mean that the author is barred from copyright protection in his work if he used earlier works. Rather an author has a valid copyright provided he has expended sufficient further independent skill and/or labour in creating his work. It follows therefore, that a program comprising a collection of known subroutines has 'originality' within the meaning of the Act if the programmer has expended sufficient skill and expertise in choosing and ordering the subroutines. Most computer programs are probably 'original' within the meaning of the Act.[25]

An Interpretation of 'Original Artistic Work'

Referring briefly to artistic works, Section 3(1) states that the class of artistic works includes:

> "... irrespective of artistic quality ... paintings, sculptures, drawings, engravings and photographs."[26]

A flowchart representing an algorithm is a drawing, and therefore is an artistic work within the meaning of the Act.

Section 3(2) states that copyright subsists in every 'original artistic work'. Therefore, provided the flowchart is the programmer's own work, then artistic copyright probably subsists. Furthermore, it has been suggested that a flowchart is analogous to a table or compilation[27] used in the definition of a literary work. Thus literary as well as artistic copyright may subsist in such works.

THE RIGHTS OF THE PROGRAM COPYRIGHT OWNER - Restricted Acts

The copyright owner of a literary work has a number of exclusive rights. These rights are defined in Section 2(5) and are termed 'the restricted acts'. For computer program copyright owners, the relevent exclusive rights are:

> "(a) [the right to] reproduc[e] the work in any material form;
> (b) [the right to] publish the work;
> .....
> (f) [the right to] mak[e] any adaptation of the work; [and]
> (g) [the right to] do in relation to an adaptation of the work, any of the acts specified in relation to the work in paragraphs (a) to (e) of this subsection."[28]

Any person doing any of these acts in relation to a copyrightable computer program without the permission of its copyright owner infringes his copyright. Infringements of these rights are termed 'primary' infringements.

A leading case outlining and establishing the copyright

owner's rights is Ladbroke (Football) Ltd. v. William Hill (Football) Ltd. (1964) 1 WLR 273. In particular, the essence of the copyright was expressed by Lord Devlin as follows:

> "The law has not found it possible to give full protection to the intangible. But it can protect the intangible in certain senses, and one of these is when it is expressed in words and print."[29]

Lord Devlin was careful not to restrict the scope of protection to the original expression of the work. However, he also considered that copyright title did not bestow upon the copyright owner a monopoly in the ideas underlying the work. Thus, the copyright in a computer program does not merely provide redress for slavish copying, but at the other extreme it cannot protect the algorithm underlying it. Rather, the rights lie somewhere in between the two, the scope of protection depending, in the final analysis, on the degree of originality of the work and the nature of the infringement.

Referring to the wording of the restricted acts, the term 'reproduction' as used in Subsection 2(5)(a) is inexhaustively defined in Section 48(1), to include, for literary works:

> "... a reproduction in the form of a record or of a cinematograph film."[30]

The term 'record' is also defined, but is limited in scope to a 'sound' recording. An exact copy of a computer program is probably 'reproduction in any material form' and therefore the making of such copies is a restricted act.

The term 'adaptation' used in Subsection 2(5)(f) is defined

in Subsection 2(6). The relevant clause therein states that 'adaptation' means 'translation'.[31] While the term 'translation' is not defined, there is nothing in the statute to limit the scope of the term to the traditional human written languages such as French and German. Therefore, translation of a computer program from one computer language to another is probably a restricted act. Furthermore, updating and enhancing a computer program may also be considered as adaptations of the work.

The scope of the restricted acts for computer programs has yet to be considered in a case argued before the Court of Apeal or the House of Lords. However, it has been considered, albeit briefly, in Sega Enterprises Ltd. v. Richards and Another 1983 FSR Part 2, pp. 73-75, heard in the Chancery Division of the High Court. In an action for interlocutory injunction, the plaintiff alleged that the defendant had infringed the copyright subsisting in a video game computer program called FROGGER, stored in an Erasable-Programmable-Read-Only-Memory (EPROM). Mr. Justice Goulding presiding, granted the plaintiff the injunction he sought, and stated:

> "On the evidence before me in this case I am clearly of the opinion that copyright under the provisions relating to literary works in the Copyright Act 1956 subsists in the assembly code program of the game 'FROGGER'. The machine code derived from it by the operation of part of the system of the computer called the assembler is to be regarded, I think, as either a reproduction or an adaptation of the assembly code program, and accordingly for the purposes of deciding this motion I find that copyright does subsist in the program."[32]

While the judge was uncertain which restricted act was applicable, he was nevertheless convinced that the plaintiff had a valid copyright in his machine-readable embodiment of the FROGGER program, and that his copyright had been infringed by the defendant.

In the aforementioned Green Paper on copyright law reform, the Government argued that the copying of programs into machine-readable embodiments is already included in the restricted acts as follows:

> "Computer programs conventionally undergo various transformations, for example from human-readable 'source code' to machine-readable digital 'object code' and vice versa, or from one source code to another. Such alternative expressions of the original programs will clearly lie within the term 'adaptation' ..."[33]

However, it is debatable whether relying upon an expansive interpretation of the term 'adaptation' in Section 2(6) provides sufficient protection. Indeed a majority recommendation of the Whitford Committee was that an explicit 'use right', an additional restricted act, was required to protect the program copyright owner from unauthorised use of his program.[34] However, the Government disagreed with this recommendation, and proposed instead that:

> "... it is sufficient for the copyright owner to have control of the initial loading of his program into the computer."[35]
> [My emphasis]

Unfortunately it is not clear what constitutes 'initial loading',

and particularly whether the unauthorised use of a program stored in a computer network would be an infringement of this proposed restricted act.

Furthermore, without case law, it is not yet established whether use of a program in a computer is included in one or more of the restricted acts, even though it involves the processes of reproduction and translation of the program from one computer language to another.[36] The main advantages of a 'use right' would be both to remove the current uncertainty and in the calculation of damages. Every act of unauthorised use of a program would be included in the calculation of the award, and therefore persistent infringers would incur high financial penalties. However, if the Government's 'initial loading' right proposal is enacted, then a guilty defendant may avoid a large sum being awarded against him if he can prove that his actions were not included within the scope of the term 'initial loading'.

It has long been established that infringement of copyright in a work can occur in circumstances where the infringer had copied some, but not all, of the work. The infringer must have copied a 'substantial part'[37] of the literary work for it to be regarded as a primary infringement under Section 2(5).

The meaning of the term 'substantial part' was considered by the House of Lords in Ladbroke (Football) Ltd. v. William Hill (Football) Ltd. (1964) 1 WLR 273. Lord Reid's comment therein

summarised the circumstances in which substantial copying is deemed to have occurred as follows:

> "... the question whether the defendant has copied a substantial part depends much more on the quality than on the quantity of what he has taken."[38]

From Lord Reid's dictum, a 'substantial part' of a work is that piece which contains the originality of the work. It follows that the amount of skill and/or labour expended in producing the 'copied' piece of the work as a proportion of the total is all important. Thus, if a computer program comprises many standard subroutines and a few original ones, then copying of the latter alone may consitute copying of a 'substantial part' of the programmer's expended skill and/or labour. If so, unauthorised copying of those subroutines constitutes an infringement of the copyright owner's rights in the program.

The Rights of the Artistic Copyright Owner

The four restricted acts for artistic works are specified in Section 3(5).[39] For algorithms expressed as flowcharts, arguably the most relevant restricted act is 'reproduction in any material form', specified in Subsection 3(5)(a). There is no equivalent provision relating to adaptations of artistic works. Thus artistic copyright provides a narrower scope of protection than literary copyright, for the infringing flowchart must not only convey the same information as the original, but must also look like the original.[40]

AUTHORSHIP OF PROGRAMS

Copyright vests primarily with the author, or in appropriate circumstances, his successor in title, or the person(s) who commissioned the work. It does not necessarily follow that the author is only the person who committed the ideas to paper or some other material form. Rather co-authorship depends on the amount of skill and/or labour contributed by each co-author. Thus for a computer program, the copyright owner(s) could be (a) the systems analyst, (b) the programmer, and (c) the coder, not to mention other possible contributors.

In the case of an employee, the ownership of copyright is less complicated, for copyright in programs developed in the course of his employment is usually assigned to the employer in his terms of engagement.

Copyright title may not necessarily be so clear cut for a freelance programmer without explicit provision in his contract.

ENFORCEMENT OF COPYRIGHT IN COURT

There are a number of conditions that need to be satisfied before a court holds that primary infringement of a literary work has occurred, or will occur.

First, the plaintiff must be the copyright owner or an exclusive licensee. If the defendant can prove that the plaintiff was not entitled to the copyright, or that he was not the only person entitled to it, the plaintiff's case collapses, even

where the defendant is clearly infringing. It is common practice in such actions for defendants to insist upon strict proof of title.[41] This defence may be particularly effective in future software infringement cases, for many programs are copied or based upon earlier ones, in which copyright title may be vested in a third party.

Secondly, there must be an 'objective similarity' between the two works. This does not necessarily mean that they must look alike, although in certain cases this would be sufficient. Rather one must be a reproduction or adaptation of the other.[42]

Lastly, there must be a 'causal connection'[43] between the two works, namely that the literary work was the source of the alleged infringer's work. This does not mean that there has to be direct derivation, for it has been held that indirect derivation (i.e., copying at second, and even third, hand) is sufficient.[44] Further, subconscious and/or inadvertent copying, if proved, is sufficient.[45] However, mere similarity between the two works does not by itself establish a causal connection, but it does provide evidence of copying.

If the plaintiff can produce, as evidence, a copy of the defendant's program or associated documentation, and show that it is identical, or substantially similar, to his own software, then this would provide persuasive evidence of copying,[46] and therefore infringement of his copyright. His case can be further strengthened if he can prove that intentional, but insignificant, mistakes and/or hidden copyright notices, contained in his program,

are replicated in the defendant's program.[47] However, a plaintiff rarely has such evidence to present before the court, and therefore the case is usually judged on the balance of probability.[48]

Various features of the plaintiff's and the defendant's programs may aid the plaintiff's case. For example, if both contained the same non-standard routine designed for the same purpose when an obvious alternative is available, then this may prove copying between two otherwise dissimilar programs.[49] The testimony of expert witnesses is particularly important in explaining to the court the similarities and differences between them, and the significance of any similarities.[50]

If the plaintiff succeeds, then the civil remedies available are an injunction, an award of damages, an order of the court for the delivery up of infringing copies, and/or an account of profits made from dealings in the infringing copies.[51]

One of the most effective remedies available at the interlocutory stage of litigation is the Anton Piller Order. This is an order of the court to search and remove alleged infringing material from the defendant's premises. The application for an Anton Piller Order is heard in camera. Given the power of the Order (for it is almost a civil search warrant), it is only granted to a plaintiff having an extremely strong prima facie case and reasonable grounds for fearing that the defendant will destroy evidence if the application is heard inter partes.[52] A further attraction of the Anton Piller Order is its speed.

The Action can be heard and the Order given in several days.

The Copyright Act 1956 provides some protection to the program copyright owner, but there are also many gaps, or uncertainties, in this protection. In particular, without statutory provision for programs or relevant case law, it is not clear whether certain machine-readable embodiments of programs are suitable subjects for inclusion in the class of protected 'literary works'. In addition, the scope of the program copyright owner's exclusive rights is not clear. The current uncertainty can be partially clarified by the ever-increasing body of overseas case law, and particularly the precedents established in recent US copyright cases. Although such cases have no precedential value in the UK, they may nevertheless have a bearing on statute and/or case law developed here.

US COPYRIGHT - A Brief Legislative History

US federal copyright law is based on Article 1, Section 8 of the US Constitution, which states that:

> "The Congress shall have power to ... promote the progress of science and useful arts, by securing for limited times to authors ... the exclusive rights to their writings ..."[53]

Congress has periodically reviewed copyright law, with the result that the 1909 Copyright Act (Title 17) was replaced on 1st January 1978 with the 1976 Copyright Act. The 1976 Act was itself amended by the 1980 Copyright Amendment Act. This latest Act implemented the recommendations of the National Commission on New Technological Uses of Copyrighted Works (CONTU).[54] CONTU was established to consider, amongst other things, the use of copyrighted works in computers, including programs.[55]

Thus, in recent years the protection of programs provided by copyright has been much discussed both in Congress and in the Computing Services Industry. In addition, recent case law has substantially clarified the scope of copyright for programs, although problems still remain in other areas.

Under the previous 1909 Act, published works capable of federal copyright protection were inexhaustively defined to include 'the writings of an author'.[56] The US Copyright Office was uncertain whether a program was a 'writing' within the meaning of the Act, but nevertheless, in May 1964, allowed the registration of certain expressions of programs under their 'rule of doubt', subject to certain conditions.[57]

There was also considerable uncertainty as to the extent of the copyright owner's rights in his programs. Amongst other exclusive rights, the copyright owner was the only person permitted to 'copy'[58] the program. However, previous case law appeared to restrict the scope of the term 'copy' to visually-perceptible reproductions. In particular, in White-Smith Publishing Co. v. Apollo 209 US 1 (1908), the Supreme Court held that a pianola roll, a mechanical device, was not an infringing copy of the same tune expressed as a copyrightable music score.[59] This case has been cited in support of the argument that copyright in a program does not extend to its machine-readable embodiments.[60]

Copyright Acts 1976 and 1980 Amendment: Suitability of Programs for Protection

US copyright law was substantially overhauled with the enactment of the Copyright Act 1976. Amongst many other reforms contained in this statute, was a clearer definition of works capable of protection as follows:

> "Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device."[61] (My emphasis)

The category of specified 'works of authorship' which may include a computer program is 'literary works'.[62] However, neither of the definitions of 'work' or 'literary work' in Section 101 (the

definitions section of the Copyright Act 1976) specifies a computer program. With the enactment of the 1980 Copyright Amendment Act, the inclusion of programs as 'works of authorship' became established by the addition of a definition of a computer program in Section 101.*

Case law has gradually extended the language, type and embodiment of programs that can be protected by copyright to include systems, as well as, applications programs expressed as source code or object code in machine-readable as well as human-readable embodiments. For example, in Tandy Corp. v. Personal Micro Computers In. 524 F. Supp. 171 (N.D. Cal. 1981) the Northern District Court in California held that a systems program, embodied in a Read-Only-Memory (ROM), designed to control the input and output of data in a microcomputer system, was an original literary work under the Copyright Act 1976.# In Williams Electronic v. Artic International 685 F. 2d. 870 (1982) a judge on the third circuit considered that a program designed to control an audio-visual display, and embodied in a ROM, was protected under both the Copyright Act 1976, and the 1980 Copyright Amendment

---

* A computer program was defined as:

> "... a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[63]

However, this case should be considered alongside Apple Computer Inc. v. Franklin Computer Corp. 545 F. Supp. 821 (E.D. Pa. 1982) in which a preliminary injunction was denied the plaintiff because the Court doubted whether an object code program embodied in a ROM was suitable subject matter for copyright protection. This decision is now on appeal.

Act.[65]

According to one US lawyer specialising in computer software cases, US courts are now beginning to accept that programs expressed in any fixed embodiment are 'works of authorship' capable of copyright protection.[66]

Exclusive Rights: Case Law

Under the Copyright Act 1976, the exclusive rights of US copyright owners include:

> "(1) [the right] to reproduce the copyrighted work in copies ...;
> (2) [the right] to prepare derivative works based upon the copyrighted work; [and]
> (3) [the right] to distribute copies ... of the copyrighted work to the public by sale or other transfer of ownership, or by rental, lease or lending ..."[67]

The term 'copy' as used in (1) and (3) above is defined in Section 101 as:

> "'Copies' are material objects ... in which a work is fixed by any method now known or later developed, and from which the work can be perceived, reproduced or otherwise communicated either directly or with the aid of a machine or device."[68] (My emphasis)

The term 'fixed', as used above, is also defined:

> "A work is 'fixed' in a tangible medium of expression when its embodiment in a copy ... is sufficiently permanent or stable to permit it to be perceived, reproduced or otherwise communicated for a period of more than transitory duration."[69]

With the enactment of the 1980 Copyright Amendment Act, the scope of a program copyright owner's exclusive rights extended to the breadth of those definitions, to include the right to reproduce the program in source code or object code, in any fixed embodiment irrespective of whether or not it (the embodiment) is human-readable or machine-readable.

Case law is gradually exploring this extended scope of the program copyright owner's exclusive rights. In GCA Corp. v. Chance (24 PTCJ 574, 7th Oct. 1982) the Court held that copyright in the source code of a program extends to its object code representation, stating:

> "... because the object code is the encryption of the copyrighted source code, the two are to be treated as one work; therefore copyright of the source code protects the object code as well."[70]

In Williams Electronic v. Artic International 685 F. 2d. 870 (1982), one of the arguments used by the defence was that a distinction had to be made between the source code and object code versions of a program. They reasoned that while the plaintiff's copyright in the source code was not in dispute, his rights did not extend to the object code translation of the source code program. They argued that the ROM embodiment of the object code program, which the plaintiff had alleged was an infringement of his copyright, was not a 'copy' as defined in Section 101 of the Copyright Act 1976. Therefore, they argued that it was not an infringement. They interpreted the term to mean that it (the copy) must:

> "... be intelligible to human beings and must be intended as a medium of communication to human beings."[71]

Since the ROM embodiment of the program is designed to be machine-readable and not human-readable, if this interpretation of 'copy' is adopted then all machine-readable embodiments of programs would then be excluded. However the Court was not impressed with this argument and stated:

> "We reject any contention that [Congress] broad language should nonetheless be interpreted in a manner which would severely limit the copyrightability of computer programs which Congress clearly intended to protect. We cannot accept defendant's suggestion that would afford an unlimited loophole by which infringement of a computer program is limited to copying of the computer program text but not to duplication of a computer program fixed on a silicon chip."[72]

The same argument was used by the defence in Tandy Corp. v. Personal Micro Computer Inc. 524 F. Supp. 171 (N.D. Cal. 1981) and was rejected by the Court. It held that a computer program was a 'work of authorship' capable of copyright protection and that a silicon chip embodiment of the program was a:

> "... tangible medium of expression ... such as to make a program fixed in that form subject to the copyright laws."[73]

These cases form part of a growing body of case law establishing that the exclusive rights of the US copyright owner extend to all fixed embodiments of the program regardless of whether they are machine-readable or human-readable. These embodiments must, from Section 101, be:

> "... sufficiently permanent ... to be perceived, reproduced or otherwise communicated for a period of more than transitory duration ..."[74]

Thus use of a program in a computer may not be included in the copyright owner's exclusive rights, for the object code is not 'fixed'. The expression of the program in the computer will probably be considered as a reproduction of transitory duration.

SUMMARY AND CONCLUSIONS

In the UK the applicability of the Copyright Act 1956 to the protection of computer programs has yet to be established. Clearly certain items of computer software have literary copyright, e.g., user manuals and technical specifications. It is probable that computer programs written on paper attract literary copyright. Furthermore, algorithms expressed as flowchart drawings probably have both literary and artistic copyright. However, with the lack of clear statutory provision and/or case law, it is not clear whether certain expressions of a computer program in an object code language embodied in certain machine readable devices (such as ROM and RAM 'chips') constitute suitable subject matter for copyright protection. In addition, the scope of the copyright owner's exclusive rights is uncertain. In particular, it is not clear whether the restricted act, 'reproduction in any material form', includes an object code machine-readable embodiment of a program whose original copyright protected embodiment was a written text of the object code. Also it is unclear just how far the restricted act of 'adaption'

extends. While it probably includes the translation of a program from one computer language to another, it is uncertain whether it also includes the enhancement, modification and use of a program in a computer.

Following recent changes in legislation, in the US a body of case law is beginning to establish the suitability of human-readable, and machine-readable, embodiments of programs as works of authorship capable of copyright protection. While the scope of the copyright owner's rights do not extend to the algorithm underlying the program, nor it seems to mere use of the program in the computer, it is becoming established that it does extend to any fixed embodiment of the program, regardless of whether it is human-readable or machine-readable.

While the protection provided by copyright for computer programs is gradually becoming clearer in the US (and also, indirectly, in the UK) many software firms are reluctant to enforce their rights. The effectiveness of copyright depends upon the willingness of copyright owners to go to law, or at least threaten to do so. However few program copyright owners are prepared to mount copyright infringement actions in the High Court except as a last resort. However, even this is not usually a realistic option for many of the smaller software firms.[75] The problems inherent in proving ownership of copyright in programs, and infringement of this copyright, together with the need to educate counsel and judges in a technology in which many are not familiar, render the option too expensive

for many such companies. Furthermore, it makes no economic sense for a copyright owner to go to law, if, in the event of winning the action, the defendant is unable to pay the award of damages and costs made against him. Thus, for all these reasons, far from providing an incentive to go to court, the uncertainties in copyright law make the prospect of an out-of-court settlement that much more attractive. Each settlement represents a lost opportunity, for the arguments are not aired, with the result that the judiciary remain substantially uneducated in the technology, and the law remains unclear. Furthermore, some software firms pay lip-service to copyright and protect their programs in other ways.

CHAPTER 3

OTHER FORMS OF LEGAL PROTECTION: PATENTS AND CONFIDENTIALITY

While copyright promises to be an effective form of protection for computer programs, it is not the only form of protection currently available to the software supplier. Additional protection can be obtained from patents, and obligations of confidentiality, the latter attaching, in each case, a duty on the computer user to keep details of the software secret. Both forms of protection are considered below.

A. PATENTS

Introduction

Unlike copyright, a patent for an invention gives the patentee a temporary monopoly right in the invention,* in exchange for publication of details of it. If, while the UK patent is in force, anybody other than the patentee, or any of his licensees, uses the invention in the UK, then he infringes the patent.[2] Furthermore, if anybody in the UK independently devises the same invention, then he also infringes the patent.[3] Thus, the patent is a most powerful intellectual property right. As a result it

---

* The maximum term of a UK patent is now 20 years following the date of filing of the patent application at the UK Patent Office, or other such date as may be prescribed.[1]

is not only sought after, but is granted only after a detailed examination of the specified invention by the Patent Office.

It has long been established in the UK and overseas that only a certain type of invention is capable of patent protection. In the Patents Act 1949[4] (now substantially replaced by the Patents Act 1977), a patentable invention was defined as:

> "... any manner of new manufacture the subject of letters patent and grant of privilege within section six of the Statute of Monopolies and any new method or process of testing applicable to the improvement or control of manufacture ..."[5]

In the Patents Act 1977, the class of patentable inventions is left undefined as such. Rather the conditions for patentability are specified.[6] Under the US Patents Act 1952,[7] inventions capable of US patent protection are defined as:

> "... any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof ..."[8]

In the prosecution of a patent application, the Patent Office of the relevant country (or jurisdiction in the case of the European Patent Office) decides whether or not the specified invention is included in subject matter capable of protection. If the invention is <u>not</u> included, then the patent application is refused, because it is deemed to be a claim to a monopoly in unpatentable subject matter. There has been considerable uncertainty in the last 15-20 years on the status of computer programs, and particularly whether certain manifestations of them

are included in the relevant definition of subject matter capable of protection. Furthermore, this uncertainty extends to industrial products and processes, inventions in which, in each case, a program is one essential element amongst others therein.

All inventions specifying programs exclusively, or including them in conjunction with other elements, are termed 'software-inventions' hereafter.

This uncertainty also extends to the courts. While the patentability of computer programs has often been considered, interpretation of the relevant definition has changed literally, case by case. Under the Patents Act 1949, the patentability of software-inventions increased following successive decisions of the Patents Appeal Tribunal. In the US, until the latest relevant Supreme Court ruling, Diamond v. Diehr (67 L.Ed. 151), previous case law can be characterised by decisions being reversed on appeal following a re-interpretation of dicta. The result was considerable confusion.

In the UK, Section 1(2)(c) of the Patents Act 1977 states that computer programs are not patentable. However, a proviso following this subsection states that this exclusion applies only to patents and patent applications relating to computer programs 'as such'.[9] The implication is that software-inventions which include programs amongst other features may be patentable. The subsection is therefore not an absolute bar to patent protection. However, the scope of this exclusion is uncertain for, as yet, no relevant cases have been reported. Therefore, in the interim,

guidance will almost certainly be derived from practice under the Patents Act 1949 and particularly the decisions of relevant cases decided under it. In addition, guidance may be derived from the US, and particularly from the Diamond v. Diehr case (67 L.Ed. 151). While this decision has no judicial weight in UK and European law, it may have an influence (albeit indirectly) on the examination of relevant patent applications in the UK and European Patent Offices.[10]

Thus, to determine the patentability of software-inventions in the UK, it is not sufficient merely to analyse the relevant sections of the Patents Act 1977. Rather relevant case law under the Patents Act 1949, the evolution of Patent Office policy following these cases, and the patentability of software-inventions under US law are also relevant.

The Patentability of Software Inventions under the Patents Act 1949

The definition of a patentable invention has always been thought, by both the Patent Office and the Courts, to exclude schemes, rules, instructions or other material comprising merely intellectual information.[11] A computer program expressed as a written source code listing was almost certainly included. Patent claims specifying nothing more than a written source code listing of the particular computer program were not allowed by the Patent Office. However it was uncertain just how far this exclusion extended. For example, were machine-readable embodiments of programs unpatentable? Were software-inventions claimed as industrial products and/or processes also unpatentable?

In each relevant case argued in an internal Patent Office Hearing, and on appeal, in the Patents Appeal Tribunal, the decision on patentability hinged upon interpretation of the term 'manner of new manufacture' used in the definition of a patentable invention.[12] If the novelty of the software-invention was established, then the question became: was it a 'manner of manufacture'?

The early cases relied upon an interpretation of the term developed by the High Court of Australia in a non software-invention case, NRDC's Application (1961) RPC No. 6; p. 135.* In this case, the Court considered that, for an invention claimed as a method or process, in order for it to be patentable an 'end product' must result from it. The Court provided examples intending to convey the scope of the term 'end product' as:

> "... any physical phenomenon in which the effect (of the method or process), be it creation or mere alteration, may be observed: a building (for example), a tract or stratum of land, an explosion, an electrical oscillation."[14]

In spite of the generality of the language used, the UK Patent Office interpreted the scope of the term narrowly. This was given some weight following Slee and Harris' Applications (1966) RPC, p. 194, the first software-invention case to be argued in the UK.

The case was heard at an internal Patent Office hearing,

---

* The relevant section (S49(2)) of the Australian Patents Act 1952-1955 contained an identical definition of patentable subject matter as that included in the UK Patents Act 1949.

following the rejection of patent applications 19463/62 and 19464/62 by the Patent Office.[15] The software-invention at issue was a method of processing simultaneous linear equations or inequalities or a mixture of both, in a computer. The invention lay in the realisation that successive iterations could be performed whilst the computer was performing the previous one (or ones). Thus, at any one time the computer would be processing a plurality of iterations simultaneously. It was emphasised in the description accompanying the original claims that the method would be useful in the design and control of industrial processes. Both inventions were claimed as methods and both patent applications were initially refused by the Patent Office on the grounds that the claimed inventions were not 'manners of new manufacture'.[16] The solution matrix, the 'end product' of the methods in each case, was considered to be merely intellectual information. The Patent Office, and the Superintending Examiner presiding the hearing, concluded that a solution matrix was outside the scope of the Australian High Court's interpretation of 'end product'.[17] However, the applications were subsequently accepted when Slee's software-invention was claimed as "a computer programmed to perform the iterative algorithm etcetera",[18] and Harris' invention as "linear programming means etcetera",[19] that is, as products. The Superintending Examiner argued that each software-invention became a manner of manufacture when claimed as a product because the programmed computer specified in the claims:

"... may be regarded as a machine which has been temporarily modified ..."[20]

The Patent Office subsequently regarded practically all software-inventions claimed as methods or processes as unpatentable.[21] However, in some cases if the same invention was instead claimed as a programmed computer, modified apparatus, or other product, then, from the Slee and Harris case, it became capable of protection. The difference between acceptable product claims and unacceptable method claims lay in the scope of the monopoly conferred. Slee's product claims covered only a programmed computer etcetera, whereas the rejected method claims specified a method of operating data processing apparatus, a potentially wider monopoly. Thus the case restricted the patent protection available for software-inventions.

The suitability of software-inventions as subject matter capable of patent protection was next considered four years later in Badger's Application (1970) RPC, p. 36. The invention at issue was a method of designing, and forming a drawing illustrating, a piping system inter-connecting several operating units. In the description of the invention, it was emphasised that the method could be used in the design of chemical plants, in which the distillation towers, storage tanks, pumps etc. could be connected together by pipes. The units were represented on a plan and the connecting pipes constrained according to certain specified rules, namely no two pipes must occupy the same space on the plan, hot and cold pipes must be at least six feet apart in real space, and all pipes must be directed along the X, Y or Z axes.

According to the method claimed, the relevant data (e.g. unit dimensions, pipe dimensions etc.) and the program (i.e. the constraints etc.) would be processed in a computer and the output converted into a drawing by means of a standard plotter.[22]

The claims were initially rejected by the Patent Office on the usual grounds that they did not specify a 'manner of new manufacture',[23] following the Slee and Harris case. In the subsequent Patent Office Hearing, the end product of the method was considered to be data representing interconnected piping lines between operating units, that is mere intellectual information.[24] The patent application was again refused. The Applicants appealed. The subsequent case heard in the Patents Appeal Tribunal was the first software-invention case to be heard outside the Patent Office. The Tribunal held that the invention was patentable as a process, if the patent claims specified a method for conditioning a computer to operate in a new way.[25]

Thus, interpretation of the term 'manner of manufacture' was now extended to include certain methods as well as software-inventions claimed as products.

Following the Badger decision, the Patent Office issued guidelines to Examiners defining those classes of software-invention now thought capable of protection. These guidelines were as follows:

> "Patents are not granted for computer programs expressed as such. No objection is, however, raised in respect of

> inventions for novel methods of programming computers to operate in a specified way, or for computers so programmed, or for a tape etc. having recorded on it a novel program to control a computer to operate in a stated way. Nor, in general, is objection taken to inventions involving new uses for computers in controlling manufacturing processes or to methods of testing, involving novel programs, for computers under manufacture."[26]

The first category, that is, 'novel methods of programming computers to operate in a specified way', followed from the Badger decision. The second and third categories followed directly from the Slee and Harris decision. However, software-inventions claimed as methods of operating computers were not included. Furthermore, in subsequent guidelines, in addition to the above, methods of operating computers were considered unpatentable subject matter.[27]

Patent Office guidelines were again revised following Burrough's Corporation (Perkins') Application (1974) RPC, p. 147.

Here, the invention at issue was a method of transmitting data between a central processor and a number of slave terminals. Each of these slave terminals was a computer. In normal operation, the central processor addressed them in a sequence controlled by the coded addresses of the terminals. If a terminal had an important message to send, it could interrupt this procedure by selectively modifying its terminal address, isolating it from incoming data from the central processor. It would then transmit its urgent message.[28]

The invention was claimed as 'a data terminal' (Claims 1-6 and 11), 'the whole system' (Claims 7 and 12) and 'a method of transmitting data' (Claims 8-10 and 13). The Patent Office originally allowed the product claims, namely Claims 1-6, 11 and 7 and 12, but objected to the method claims (Claims 8-10 and 13) on the grounds that they did not relate to a 'manner of manufacture'.[29] The subsequent Patent Office Hearing upheld the objection arguing that the end product was a different address for the terminal, merely intellectual information.[30] The applicants appealed, and in the course of the subsequent hearing before the Patents Appeal Tribunal, the Patent Office policy concerning the examination of relevant patent applications was expressed as follows:

> "... the present Office view is that a line, albeit a thin one, can be drawn between claims to 'methods of programming a computer' on the one hand and, for example, claims for methods 'of transmitting data' or 'for controlling a computer' on the other, on the basis that the product of the former is a computer programmed in a particular way while the product of the two latter is merely intellectual information. The desirability of such a line being drawn is said to be that the Office will otherwise be logically compelled to allow claims which will, for example, monopolise simple methods of operating the buttons of an ordinary office calculator, and even, possibly, methods for the solution by a schoolmaster of quadratic equations."[31]

The Court considered that methods of programming a computer and methods of controlling a computer were identical and that the

distinction made by the Patent Office was without substance. As a result the Court extended the patentability of software-inventions by stating:

> "If a claim, whatever words are used, namely, whether the claim is, for example for 'a method of transmitting data ...', 'a method of controlling a system of computers' or 'a method of operating or programming a computer ...', is clearly directed to a method of involving the use of apparatus modified or programmed to operate in a new way, ... it should be accepted."[33]

For the method claims at issue, the Court decided that the method was embodied in the apparatus and resulted in the apparatus operating in a new way.[34] The claims were allowed.

Thus the patentability of software-inventions was extended to new methods of operating computers. Patent Office policy was revised accordingly.[35] This decision was underlined by the Patents Court* in IBM's Application (1980) FSR, p. 564.[36]

Under the Patents Act 1949, the patentability of software-inventions hinged upon interpretations of the term 'manner of new manufacture'. Following the Slee and Harris case, software-inventions claimed as methods or processes were excluded from protection because in each case the 'end product' was thought not to be 'a manner of new manufacture'. However, software-inventions

---

* From Schedule 4, paragraphs 11(3) and (4) of the Transitional Provisions of the 1977 Act (Appeals of the comptroller under continuing provisions of 1949 Act) p. 120, reference to the Patents Court means reference to the Patents Appeal Tribunal.

specified as 'a programmed computer ...' or 'programming means ...', that is, in product claims, passed, and thus these inventions were capable of protection. This narrow interpretation of the term began to crumble following Badger's Application, in which the Patents Appeal Tribunal held that a software-invention claimed as a 'process for conditioning a computer' was patentable. Novel methods of 'operating computers' were held to be patentable following Burroughs Corporation (Perkins') Application, a decision underlined by the Patents Court in IBM's Application. Thus many new software-invention claimed as an industrial product, method and/or process were now capable of patent protection.

With the substantial repeal of the Patents Act 1949 on 31st May 1978, it is uncertain whether its case law is applicable to patent applications prosecuted under the new legislation, the Patent Act 1977.

The Patentability of Software-inventions under the Patents Act 1977

Unlike the Patents Act 1949, a patentable invention is not defined in the Patents Act 1977. Rather four requirements for patentability are specified. These are:

(1) the invention must be new;

(2) it must involve an inventive step, meaning that it must not be obvious to a person skilled in the Art;[37]

(3) it must be capable of industrial application; and

(4) it must not fall within a class of inventions which are excluded from patent protection.[38]

An invention must have all these characteristics in order to be patentable. Some software-inventions have requirements 1-3, but computer programs are included in the excluded class of inventions.[39] However, a curious proviso following the exclusion states that the exclusion applies only if the patent or patent application '... relates to that thing as such'.[40] In the absence of new Patent Office guidelines and relevant case law, it is uncertain just how far this exclusion extends for software-inventions.

One of the aims of the Patents Act 1977 is to harmonise UK Patent Law with the European Patent Convention (EPC).[41] Indeed, the above requirements for patentability, including the exclusion and the proviso, were drafted so that they would have the same effect as Article 52, the corresponding provisions in the EPC.[42] Thus the patentability of software-inventions under the Patents Act 1977 depends upon practice under the EPC.

In interpreting Article 52, European Patent Office Guidelines include the following:

> "If the contribution to the known art resides solely in a computer program then the subject-matter is not patentable in whatever manner it may be presented in the claims. For example, a claim to a computer characterised by having the particular program stored in its memory or to a process for operating a computer under control of the program would be as objectionable as a claim to the program per se or the program when recorded on magnetic tape."[43]

The breadth of the exclusion contained in the guidelines is

contrary to the liberal interpretation of patentable subject matter developed in case law under the Patents Act 1949. However, in practise, it seems that each European patent application is considered on its own merits, and that the exclusion is not as broad as that suggested in its wording.[44]

Many computer programs are based upon published prior art. It is thought that many software-inventions would not be patentable in the UK, because each has insufficient inventive weight to support a patent application (requirement 2). Indeed, it was estimated that as few as 1% of programs developed are sufficiently inventive for the purposes of the Patents Act 1977.[45]

There currently exists considerable uncertainty concerning the patentability of software-inventions under both the Patents Act 1977 and the EPC. Until a body of relevant case law is established, guidance is derived from cases heard under the previous legislation, and also indirectly, from foreign case law. Although US case law is not directly applicable, there has been, over the years, a large number of software-invention cases heard in the US. Guidance is probably derived from the arguments expressed in, if not the dicta established from, such cases.

Patentability of Software-inventions under the US Patents Act 1952

Until the Diamond v. Diehr Case (67 L.Ed. 155) and subsequent US Patent and Trademark Office guidelines,[46] there was considerable uncertainty concerning the patentability of software-inventions. In practically every case concerning a software-invention to date,

the Court's decision hinged upon interpretations of Sections 101 and 100(b) of the Patents Act 1952, specifying subject matter capable of protection. If the claimed invention was regarded as: "... a process, machine, manufacture, composition of matter or any new and useful improvement thereof ..."[47] it was capable of protection. If it was not included in one or more of these categories, it was deemed non-statutory subject matter and the invention was prima facie unpatentable. Interpretation of the scope of these categories, and particularly 'process',[48] changed following each case. The result was confusion for it was uncertain whether decisions of the US Patent Office would be upheld, on appeal, in the Court of Customs and Patents Appeals. The Diamond v. Diehr case clarified previous Supreme Court dicta. Furthermore, together with a new Patent Office administration, it markedly changed Patent Office policy towards software-inventions. Thus the importance of the Diehr decision appears only after previous cases and Patent Office practice have first been considered.

In common with UK practice, ideas, mental concepts, mathematical formulae and laws of nature, when claimed as such, have always been considered non-statutory subject matter, and therefore unpatentable.* Alternatively, inventions which can only be

---

* For example, in In re. Abrams (188 F.2d. 165; 89 USPQ 266; CCPA 1951) the Court of Customs and Patent Appeals (CCPA) held:

> "Citation of authority in support of the principle that claims to mental concepts which constitute the very substance of an alleged invention are not patentable is unnecessary. It is self evident that thought is not patentable."[49]

performed using physical apparatus have always been considered statutory matter, and therefore capable of patent protection.[50] But the status of inventions that can be implemented either physically or mentally was thought to be less clear cut. The US Patent Office originally regarded computer programs as inventions of this dual mental/physical nature (even though every practical implementation of the algorithm underlying a program would be in conditioning or controlling electronic circuits in a computer, i.e., physical). Furthermore this attitude towards software-inventions extended to those claimed as industrial products and computer-implemented industrial processes.

In an attempt to distinguish statutory software-inventions from non-statutory ones, the US Patent Office devised a test known as the 'mental steps doctrine'.[51] According to this test, a software-invention specified in the patent claims would be dissected into 'mental' and 'physical' elements. The invention was considered non-statutory matter if its inventive step (i.e. the essence of the invention) resided solely in the mental element.[52] Using this test, the US Patent Office regarded claims specifying software-inventions as claims to mathematical algorithms. Software-inventions were deemed non-statutory subject matter, because dicta concerning the exclusion of mathematical formulae, and the like, from patent protection were thought to apply.[53] Thus patent applications specifying software-inventions were rejected.

However the Court of Customs and Patent Appeals (CCPA)

consistently held a more liberal interpretation of Sections 101 and 100(b), subject matter capable of protection. In In re. Prater and Wei (152 USPQ 583, CCPA 1968) the first significant software-invention case, the CCPA held that a method of processing spectrographic data was patentable, provided the invention was specified in product claims. The applicants' method claims were rejected, but for reasons not connected with Sections 101 and 100(b).[54] The Court did not exclude the possibility of patentable software-inventions specified as methods. Also, in a subsequent case, it described the US Patent Office use of the mental steps doctrine for determining the status of software-inventions claimed as methods as 'inappropriate and irrelevant'.[55]

However, the mental steps doctrine was substantially re-established following Gottschalk v. Benson (34 L.Ed. 273). This was the first case concerning the patentability of a software-invention to be heard by the US Supreme Court.

The alleged invention was a method of converting base 10 numbers into binary numbers in a digital computer. The inventive step resided solely with the program used in the computer. The US Patent Office originally rejected the method claims on the usual grounds that they specified non-statutory matter, a decision reversed in the subsequent CCPA hearing.[56]

The Supreme Court considered that the case turned upon whether the patent claims specified merely an algorithm, or 'an application of the law of nature to a new and useful end'.[57] If the claims specified the former the invention was non-statutory

matter following established practice concerning the exclusion of mathematical formulae, laws of nature, and the like. On the other hand, if it specified the latter then it became statutory matter capable of protection. Unfortunately the Court failed to answer its own question for it upheld the Patent Office objection stating:

> "It is conceded that one may not patent an idea. But in practical effect that would be the result if the formula for converting BCD numberals to pure binary numerals were patented in this case. The mathematical formula involved here has no substantial practical application except in connection with a digital computer, which means that if the [CCPA] judgement below is affirmed, the patent would wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself."[58]

The Court conceded that the claims specified a method requiring a digital computer, something more than merely a recitation of Benson's algorithm, but this did not prevent the invention from being deemed non-statutory subject matter.

The next significant case before the Supreme Court was Parker v. Flook (57 L.Ed. 2d. 451). Flook's invention was a method of updating the value of an alarm limit in the catalytic chemical conversion of hydrocarbons in oil refining. It comprised three steps:

(i) measuring the present value of the variable;

(ii) calculating an updated value of the alarm limit using a programmed computer; and

(iii) adjusting the actual alarm limit to the updated value.

The inventive step resided solely in the program used in the second step of the method.[53] The Patent Office rejected the method claims arguing that the sole difference between the claimed invention and the prior art lay in the mathematical algorithm of the program, citing Gottschalk v. Benson.[60] The CCPA, on the other hand, interpreted the dicta from Gottschalk v. Benson as applying only to claims that entirely pre-empted a mathematical algorithm. It argued that the scope of the claims was limited to a specified industrial process and that the Benson dicta were irrevelant. The court reversed the Patent Office decision and allowed the method claims.[61] On appeal, the Supreme Court considered the matter at issue to be whether Flook's method described a 'process' within the meaning of the Patent Act,[62] and therefore, as noted by the Court, the "... case turns entirely on the proper construction of section 101."[63] In a split decision, 6:3, the majority held:

> "... the method of updating alarm limits was not patentable under section 101 ..., the identification of a limited category of useful, though conventional post-solution applications of the formula not making the method eligible for patent protection."[64]

The Parker v. Flook case established that, for a software-invention claimed as a method, something more than merely an updated alarm limit as an end product was required in order for the invention to be capable of protection. However, the Court did not specify what was required. It also failed to clarify the

scope of its previous Gottschalk v. Benson dicta. Both of these uncertainties were substantially dispelled following the latest software-invention case argued in the Supreme Court, Diamond v. Diehr (67 L.Ed. 155).

The invention at issue was a method of curing synthetic rubber in a press. According to the prior art, the cure time depended on the temperature of the mold, and since this was variable over time, perfectly cured molds could not be guaranteed; temperature was an uncontrollable variable. The invention comprised monitoring the temperature inside the mold, continuously updating the cure time by repeatedly solving the cure time equation in a programmed computer, and once the cure time equalled the actual time the mold had been in the press, automatically opening it.[65] In this way perfect cures could be ensured.

The US Patent Office rejected Diehr's method claims, on the grounds that they specified a program, citing Gottschalk v. Benson. This decision was upheld by the Patent Office Board of Appeals, but reversed in the CCPA hearing.[66]

The Supreme Court upheld the CCPA decision, arguing that the invention was an industrial process capable of protection and not merely a recitation of a mathematical algorithm.[67] Furthermore, the Court rejected the claim dissection techique, inherent in the mental steps doctrine.[68]

Following the Diehr decision, the US Patent Office issued new guidelines to examiners.[69] The guidelines specified a

two-stage test* to determine the status of software-inventions under Section 101. According to the first stage:

> "... each method or apparatus claim must be analysed to determine whether a mathematical algorithm is either directly or indirectly recited. If the claim at issue fails to directly recite a mathematical algorithm, reference must be made to the specification in order to determine whether claim language indirectly recites mathematical calculations, formulas or equations.
>
> If a given claim directly or indirectly recites a mathematical algorithm, the second step of the analysis must be applied."[71]

The 'indirect recitation' of a mathematical algorithm includes a computer program when claimed as such, or in claims reciting certain industrial products and computer-implemented industrial processes. Thus all software-inventions will require the second stage of the test. According to this stage:

> "... a determination must be made as to whether the claim as a whole, including all its steps or apparatus elements, merely recites a mathematical algorithm or method of calculation. If so, the claim does not recite statutory subject matter under 35 USC101."[72]
> (Emphasis is mine)

From these guidelines, the inclusion of a computer program in a software-invention no longer attracts an initial Patent Office rejection of the patent application on the grounds that

---

* This test is derived from CCPA dicta developed in two cases. The test was originally devised in In re. Freeman 573 F.2d. 1293, 197 USPQ 464 (CCPA 1978) and modified in In re. Walter 618 F.2d. 758, 205 USPQ 397 (CCPA 1980).[70]

the subject matter claimed is non-statutory. However, the claimed invention must specify something more than a computer program limited to a general technical field in order to avoid rejection of the patent application under Section 101.[73] Thus, if the claimed software-invention is limited to a specific technical field, then from the second stage, it will probably be regarded as subject matter included in Section 101, and thus will be capable of patent protection.

Summary and Conclusions

The suitability of software-inventions for patent protection is uncertain under the UK Patents Act 1977, the European Patent Convention, and to a lesser extent, the US Patent Act 1952. In the UK and Europe, it is established that a computer program, when claimed as such, is not patentable, because it is included in a specified class of inventions excluded from protection. Thus a program claimed as a written source code listing, without restriction to a specific industrial application is unpatentable. However, it is uncertain just how far this exclusion extends.

Patent examiners and the judiciary have, over the years, struggled to understand the nature of computer programs. Originally, in whatever embodiment they were claimed, programs were regarded merely as mathematical algorithms or laws of nature. If they were not regarded as such, then patent applications for software-inventions were nevertheless rejected on the same grounds. Gradually this false analogy was realised, and

the patentability for software-inventions increased.

Under the Patents Act 1949, originally only software-inventions claimed as products were thought capable of protection. However, from case law it eventually became established that patentability extended to software-inventions claimed as industrial methods or processes. However, with the enactment of the Patents Act 1977 and the EPC, it is uncertain whether software-inventions claimed as industrial products, methods or processes are still subject matter capable of patent protection in the UK.

In the US, originally most software-inventions were regarded as non-statutory subject matter and therefore were barred from patent protection. However, from case law, it initially became established that software-inventions claimed as products were capable of protection. The subsequent interpretation of Section 101, defining subject matter capable of protection, evolved through a process of trial and error characterised by the reversal of decisions on appeal, and the examination and re-examination of Supreme Court dicta. Case law eventually established that subject matter capable of protection includes software-inventions claimed as methods or processes as well as products provided the claimed monopoly is limited to a specific application and is not merely a claim to the algorithm underlying the program.

B CONFIDENTIALITY

Introduction

Unlike patents and copyrights, the modern action for breach of confidence is not based on any property right the supplier may have in his software.[74] Neither is it based on any contractual relationship between the discloser of information and the recipient.[75] It is now generally accepted that the action rests upon an obligation on both parties to be of 'good faith'.[76] The modern action has been described generally as providing:

> "... a civil remedy affording protection against unauthorised disclosure or use of information which is of confidential nature and which has been entrusted to a person in circumstances which impose an obligation to respect its confidentiality."[77]

However, it has an uncertain basis in law*, although this, in itself, does not seem to have deterred people from using it.

The types of information that have been successfully protected are diverse, including information of an industrial, commercial and personal nature, and even information of political significance.[79] The problem of assigning a legal basis to the action is caused partly by the thing it protects, namely intangible confidential information.[80] Thus, although there have been

---

* On numerous occasions over the last 130 years the basis of the action has been considered by the courts. Although it has its origins in Equity, at one time or another, it has been held to be based upon property, contract, bailment, trust, fiduciary relationship, good faith, unjust enrichment or a combination of these grounds.[78]

no reported cases concerning software, there is nothing to prevent it from being applicable to computer programs and other valuable information of the software supplier. Indeed, in his book Computer Law, Colin Tapper regards obligations of confidentiality as:

> "... the most widely adopted method of seeking protection for computer programmes at present."[81]

Currently, the action has no statutory basis.* The case that is generally considered to best exemplify the modern action is Coco v. A.N. Clark (Engineers) Ltd. (1969) RPC, p. 41. In hearing an application for an interlocutory injunction, the presiding judge, Megarry J., isolated three elements which he considered were required (apart from contract) for the plaintiff's case to succeed:

> "First, the information itself ... must 'have the necessary quality of confidence about it.'[82] Secondly, that information must have been imparted in circumstances importing an obligation of confidence. Thirdly, there must be an unauthorised use of that information to the detriment of the party communicating it."[83]

These elements were approved by the Court of Appeal in Dunford and Elliot v. Johnston (1978) FSR 143 and Jarman and

---

* The main recommendation of the Law Commission Report on breach of confidence (HMSO, Oct. 1981, Cmnd. 8388, Part VII, pp. 169-178) was that the law should be given a statutory basis. Thus, if this recommendation is adopted, then the law will no longer rely merely on case law, rather there would be a statutory tort for breach of confidence.

Platt v. Barget (1977) FSR 260.[84]

Information capable of Protection

It has long been established that confidential information must not be 'public property or public knowledge'.[85] However, this does not mean that information, comprising known parts, is not protectable by the action. Rather it is the information as a whole which is important.[86] Thus, an obvious program comprising a collection of well-known, standard subroutines could be information capable of protection in a breach of confidence action. However the action does not protect mere private 'tittle tattle'[87] or information having insufficient originality to qualify for copyright protection.[88]

Circumstances importing an Obligation of Confidentiality

In general, it is not possible to predict when an obligation of confidentiality arises, for it entirely depends upon the individual circumstances. However, in each case, the recipient is bound by the duty only if he accepts it, or when it can be proved that he should have known in the circumstances that he was the recipient of confidential information.[89]

In certain situations, the obligation can arise from the relationship between the parties, independent of the information itself. For example, it has long been accepted that partners in a business have a mutual obligation to one another.[90] Further examples include a craftsman to his customers.[91]

It has also long been accepted that a special case exists when an employee is the recipient of his employer's confidential information.[92] In most master/servant cases the courts have attempted to strike a balance between, on the one hand, the worker's desire to work when and where he pleases and, on the other, the employer's desire to keep his confidential information secret.[93] In general, the balance rests largely with the employer during employment, and with the employee thereafter.[94]

A major problem lies in distinguishing information entrusted to the employee by the employer (information which he would not be entitled to use in subsequent employment) from information which he acquired on the job (constituting his own personal knowledge which he would be entitled to use subsequently).[95] For example, a programmer who was sent by his employer on a course in BASIC programming would be entitled to use the programming skills acquired from attending the course, in future employment. However, he would be in breach of his duty to his employer if he disclosed, or used, the employer's BASIC programs, in subsequent employment.[96] However, between these two extremes there is a wide grey area.

Whilst in his employment, it is well established that an employee has a duty of fidelity to his employer, even in the absence of a specific confidentiality clause in his terms of engagement.[97] This duty has been interpreted to include:

(a) a prohibition on an employee to work for a competitor in his spare time;[98]

(b) a prohibition to reveal his employer's secrets to his trade union;[99]

(c) a prohibition to allow employees of rival firms access to confidential information;[100]

(d) not to deliberately store information during employment for use after having left his employer's service;[101]

(e) not to use confidential information for any purpose other than the one intended;[102] and

(f) not to memorise information for use after employment.[103]

Once the programmer has left he is substantially free to use all the information gained at the expense of his ex-employer. There are, however, two exceptions. First, information that is strictly the trade secrets of his ex-employer and secondly, information concerning the goodwill between the ex-employer and his customers.[104]

These exceptions were originally identified in cases concerning restrictive employment covenants. The courts have taken the view that all covenants resulting in restraints of trade (e.g., limiting the mobility of personnel from one company to another) are against the public interest and prima facie void, unless they are reasonable.[105] By 'reasonable' it has become established that a covenant must be necessary for the protection of the ex-employer's business, the burden of proof lying with the ex-employer.[106] In addition, it must relate to the particular trade secret(s) known by the particular employee (no general or vague references are allowed), and be limited in

time, geographical area, or business activity to within reasonable limits.[107]

Unauthorised Use of Information

In its broadest interpretation, the obligation of confidentiality is a duty on the recipient not to use, or disclose, the information without the consent of the person to whom the duty is owed,[108] for any purpose additional to the one for which it was originally revealed.[109] Thus a computer user who discloses details of his software to a competitor of the company supplying him with it, is in breach of his duty to the supplier. Furthermore, if the competitor subsequently discloses or uses the software when he knew it was divulged in breach of confidence (or when he should have known, in the circumstances, that this was so) then he also is in breach of his duty to the original supplier. However, both are liable only if the unauthorised disclosure or use is deemed by the court to be detrimental to the supplier. This is not difficult to prove, particularly if the supplier incurred a loss of revenue following the unauthorised disclosure.

Remedies

All remedies are restitutionary, and not punitive, in nature. Remedies are discretionary, and thus the choice of one, or a selection, of them depends primarily upon the facts of the particular case. There are, in general, four remedies that a court could grant the successful plaintiff; an injunction, an

award of damages, an account of profits and an order for the delivery up of material containing the information.[110] There are a few tests to govern the circumstances in which each is considered appropriate.

Conclusions

The principal attraction of the action for breach of confidence for the software firm is its applicability to the protection of software. Although there have been no reported cases concerning software, an obligation of confidentiality can protect, not only the computer program, but also all associated confidentiality information.

In an action, the plaintiff must prove: (1) that the information at issue was confidential; (2) that it was disclosed to the defendant (either directly or indirectly) in confidence; and (3) that the defendant's unauthorised disclosure or use of it was detrimental to the plaintiff. For instance, the last of these conditions is satisfied if the plaintiff can prove he lost business from the unauthorised disclosure or use. However, it may not be easy to prove the existence of the remaining conditions for mass market microcomputer software. A confidentiality clause in a software licence does not, in itself, establish that the software is confidential. Referring to the second condition, since many microcomputer software products can be purchased, literally, over the counter, a court would be unwilling to hold that such a transaction imports an obligation of confidentiality on the

customer. Clearly the existence of a secrecy clause as a condition of 'sale', may provide proof that the software was disclosed to the customer in confidence. However, in general, the enforcement of the obligation in court is neither cheap nor easy. Furthermore, prediction of the court's decision is hazardous for it depends entirely upon the information at issue and the individual circumstances of its disclosure. Nevertheless it can be a valuable addition to copyright. In particular, it can be useful in protecting the algorithm and 'know-how' underlying the program which are not necessarily protected by copyright as such.

CHAPTER 4

A SURVEY OF THE INDUSTRY

INTRODUCTION

Although the scope of protection provided by intellectual property and allied rights is uncertain, this does not necessarily imply that such rights are ineffective. Indeed it may be that licences, patents, copyrights and confidentiality, together provide sufficient and effective protection for most software suppliers. However, the opinions of executives within the Computing Services Industry (arguably the people most interested in the protection of software) on the effectiveness of the law, are, for the most part unknown. Furthermore, the protection policies of software firms, and particularly those companies supplying microcomputer software, are unknown. Until now, there had been no comprehensive survey of the industry concerning software protection.

However, there have been a number of previous studies into certain aspects of software protection. In particular, in 1975 Margaret Anderson of the University of Kent conducted a survey of the industry as part of a research project on the legal protection of proprietary software, commissioned and funded by the Department of Industry.[1] Her aim was to determine the awareness of the industry to protection provided by patents, copyrights, licences etc., legal rights exclusively. Also, the US National Commission of New Technological Uses of Copyrighted Works (termed CONTU

hereafter) commissioned a study to determine whether or not legal protection affected the utilisation of software innovations in the US. The researchers (Harbridge House Inc.) reported their findings in 1978.[2] This study is the most widely reported survey on software protection, primarily because of its large sample, for 116 companies participated. However, in both studies it was assumed that legal rights in the software were the main forms of protection available. Non-legal protection measures, e.g. technical protection measures incorporated into programs, were either mentioned in passing or not considered relevant to the task at hand. Indeed, it may be that legal rights were the main forms of protection to software suppliers in the mid-1970s. However this is not the case now.

In the present survey, the results of which are reported hereafter, the protection of programs was not considered to be exclusively a legal issue. Protection measures can be written into programs. In addition, protection can stem from the inherent nature of certain programs; for example, those requiring regular maintenance and updating. Questions on each of these forms of protection were included.

The objective of the survey was to obtain information of industry practices in the protection of programs from a variety of threats, and particularly private unauthorised copying and use, and software piracy. The aims of the survey were to:

(a) determine whether or not effective protection stemmed from (i) end user's need for regular maintenance of programs, and (ii) the investment of time, money and expertise embodied in

a program;

(b) determine the number and distribution of companies that have suffered from unauthorised copying and piracy of programs;

(c) obtain opinions on the effectiveness of legal rights in the protection of programs; and

(d) obtain information on the types of technical measures used.

The survey comprised a postal questionnaire to three hundred and fifty-one firms in the computing services industry, and follow-up interviews of some of the respondents. The questionnaires were sent in April/May 1982.

Each questionnaire was divided into five sections. The first section (Questions 1-3) concerned respondents' experiences of unauthorised copying and piracy of the firms' programs. The second section (Questions 4-5) concerned the modes of legal protection used, and their perceived effectiveness. The third section (Questions 6-7) canvassed opinions on the effectiveness, as modes of protection, of two inherent features of most, if not all, programs. The fourth section (Questions 8-12) concerned technical measures used to protect programs. The fifth, and final, section (Questions 13-20) concerned characteristics of the type of company. Opinions were also canvassed, in this section, on other software protection issues of indirect relevance to firms' protection policies.

The questionnaire is reproduced as an appendix.

The names and addresses of the population of companies were

obtained from a variety of sources, including the 1981 Directory of Members of the Computing Services Association, the 1982-3 Buyers' Guide of the Computer Retailers Association, software companies advertising in the February 1982 edition of "Practical Computing" in its Software Buyers' Guide, software companies advertising in the March 1982 edition of "Personal Computer World" in its Software Buyers' Guide, Members of the Technology of Software Protection Specialist Group of the British Computer Society and Committee Members of the European Computer Manufacturers Association.

One hundred and fifty-six completed questionnaires were returned, a response rate of 44%. Two respondents represented US companies exclusively, the remaining 154 representing either UK companies or international companies each having a UK subsidiary. The questionnaires were filled in by the Managing Director, Partner, Software Development Manager, Software Sales Manager or Legal Counsel as the case may be, of the companies in the sample. Seventy-four of the 154 UK firms were subsequently contacted, respondents being interviewed mainly on the telephone,* although 16 companies were also visited.#

---

* Information was given on the understanding that its source would not be divulged. Each number represents a particular company in the sample:

007, 008, 010, 012, 016, 018, 019, 022, 023, 024, 025, 030, 031, 036, 039, 040, 043, 044, 045, 047, 050, 051, 052, 053, 056, 062, 063, 066, 068, 070, 075, 076, 080, 082, 083, 084, 086, 088, 089, 090, 093, 094, 098, 100, 106, 109, 110, 115, 117, 121, 124, 130, 131, 132, 133, 139, 140, 142.

\# 001, 002, 009, 011, 027, 032, 087, 096, 103, 118, 120, 125, 126, 127, 147, 150.

The number of companies in the sample, expressed as a percentage of the total number of companies in the UK computing services industry, is unknown, primarily because it (the industry) is extremely heterogeneous.[3] Estimates of the number of companies depend upon which types of company are included.* The 154 UK companies in the sample probably account for not less than 13% of all companies in the industry. The sample was also heterogeneous, for replies were received from companies which, as a group, supplied many different types of program, or service. They also varied greatly in size. Forty-eight firms (30.8%) had between one to nine professional or technical employees, whereas 34 (21.8%) had over 100 such employees. However, in spite of the diversity of companies in the sample, many had a common interest, namely the protection of their software.

Data from the questionnaire replies were analysed using the Statistical Package for the Social Sciences (SPSS), Version 8, on a 'Harris 500' computer at the Computer Centre, University of Aston.

---

* The 1982 COMPUTER USERS YEARBOOK lists 1115 firms including computer hardware and computer software firms. COMPUTING MARKETPLACE: A DIRECTORY OF COMPUTING SERVICES AND SOFTWARE SUPPLIERS FOR WORD PROCESSORS, MICRO, MINIS AND MAINFRAMES 1981 lists 459 software and systems houses. It also lists 181 microcomputer system retailers, many of which are included in the previous software systems houses list.

CHARACTERISTICS OF THE SAMPLE

Given the many different types of company within the industry, each respondent was asked which of the following describes his/her company:

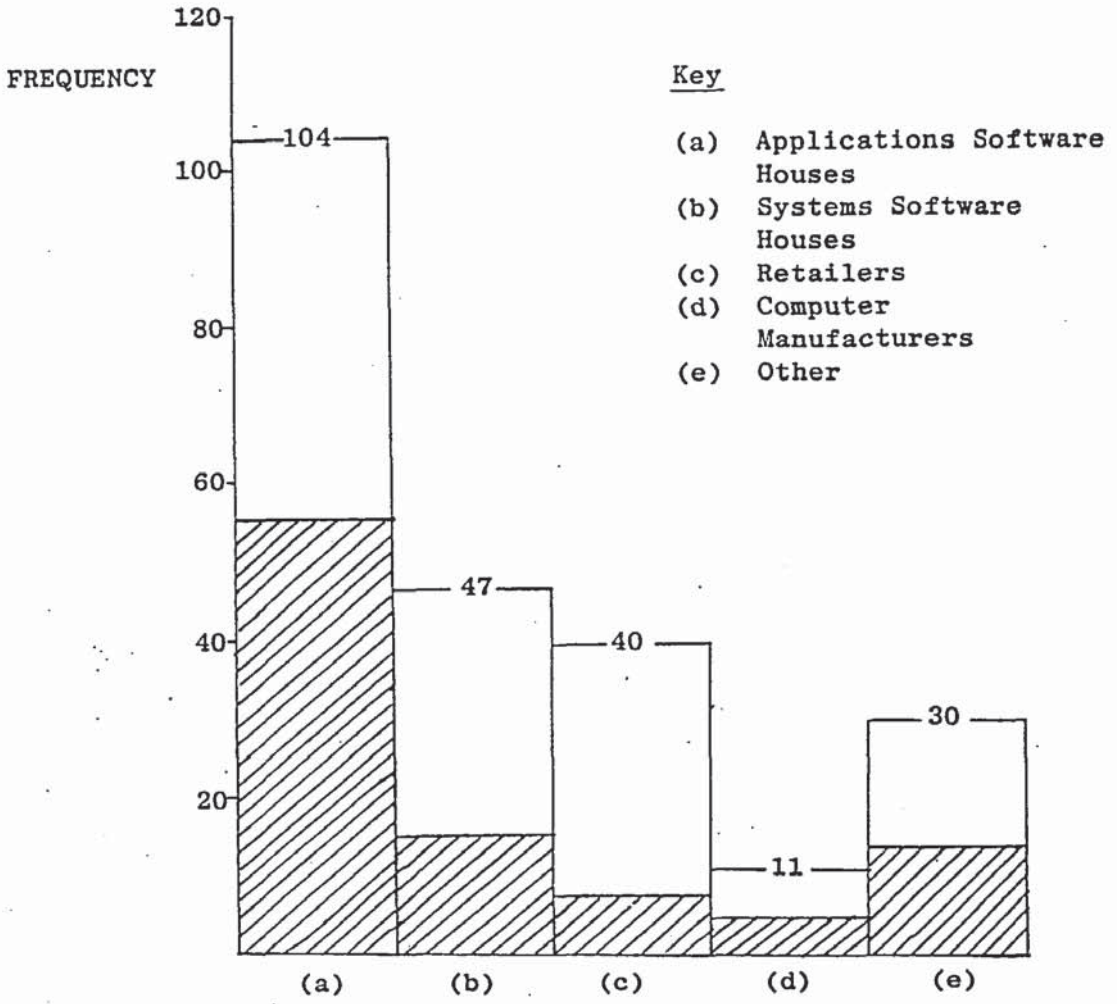
(a) An Applications Software House
(b) A Systems Software House
(c) A Software Publisher
(d) A Software Distributor
(e) A Computer Manufacturer
(f) Other. Please specify .....

All 156 respondents replied, their answers being distributed as shown in Figure 4.1 overleaf. Fifty-seven ringed two or more of the descriptions. The remaining 99 respondents used a single description, and the distribution of these replies are shaded on the figure.

Thus 104, out of 156 respondents, described their companies as 'Applications Software Houses'. This was by far the most popular description. For 56 companies, their major activity was the development of applications programs, for the single description was used exclusively. The remaining 48 respondents specified it, together with one or more of the other descriptions. Twenty-eight of these were either software publishers, distributors or both, that is firms involved in the marketing of programs. Software publishers and distributors have been combined to form a single 'retailers' category. The relevant twenty-eight companies in the unshaded portion of the 'Applications Software Houses' bar are included in the corresponding portion of the 'retailers' bar. Thus over a quarter of applications software houses in the sample

market, as well as develop, their own programs.

Figure 4.1: Description of Firms



Forty-seven respondents specified 'Systems Software Houses'. For 16 companies the development of systems software was their major activity. The remaining 31 respondents included one or more of the other descriptions. Companies in the 'other' bar were mainly computer bureaux and hardware/software consultancies. 'Computer manufacturer' was the least used description, with only 11 companies being inclusively, or exclusively, described as such.

For 57 companies, over one-third of the sample, a single traditional description does not adequately describe their activities. These companies have branched out into different, but nevertheless related, areas. The replies did not identify the major interest of these companies. Thus, firms in the sample are referred to simply by the generic term 'software firm' or 'software supplier' hereafter.

A second characteristic of the sample was the type of program supplied, or service provided. Each respondent was asked to specify their firms' products or services, according to the following general descriptions:

(a) General Business Applications (e.g., Accounting, Word Processing)
(b) General Financial Applications (e.g., Payroll, Tax)
(c) Engineering and Scientific Applications
(d) Systems Software
(e) Other. Please specify .....

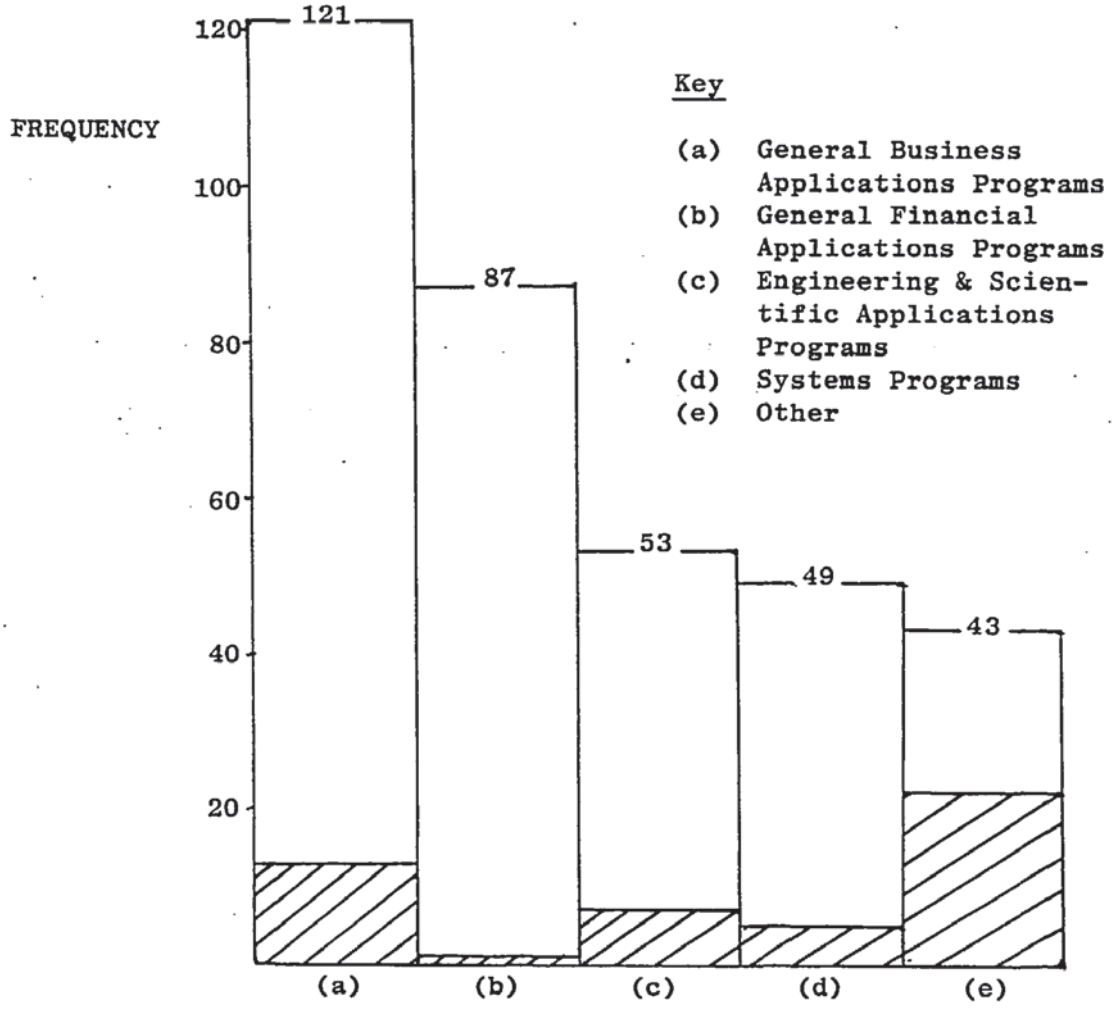
All 156 respondents replied, the distribution of their answers being represented as a bar chart in Figure 4.2 overleaf.

Only 48 firms supplied one type of program, or provided a single service, to computer users. The distribution of these replies are shaded in the relevant bars of Figure 4.2. The largest minority of these were 22 computer bureaux, consultancies or hardware suppliers comprising the shaded portion of the 'Other' bar.

The remaining 108 companies in the sample supplied more than one type of software. Thus, 121 companies out of 156 supplied 'General Business Applications' software, and for 13

this comprised their whole software product range. Eighty-seven, out of 156, supplied 'General Financial Applications' software. Only one of these companies specialised in this software to the exclusion of other types of software; and so on. Thirty firms in the sample supplied both 'General Financial Applications' and 'General Business Applications' software, and these are included in both the relevant unshaded bars of the figure.

Figure 4.2: Types of Program Supplied



Only a small handful of respondents, representing companies in the 'Other' bar, specified games and other personal software. The sample comprised companies which were, in the main, supplying software to industry, business and commerce. Furthermore, 126 respondents regarded their firms' most important markets as either large corporations, or medium to small companies. Few considered computer hobbyists and other home computer users (as opposed to business computer users) as their main markets.

The third, and arguably most important, characteristic of the sample was the compatible type of computer hardware. Respondents were asked to indicate the general type of hardware for which their software was designed. Each type of 'microcomputer', 'minicomputer' and 'mainframe computer' has its own technical limitations (e.g., memory size, processing speed) depending upon the state of the art and the market for the hardware. Given the variety of computers available, technical indicia were not used to differentiate hardware.* Rather they were distinguished from one another by retail price per unit according to the following scale:

(a) up to £10,000
(b) £10,000 - £30,000
(c) £30,000 - £80,000
(d) £80,000+

Category (a) comprises hardware generally described by the generic term 'microcomputer'; category (b) comprises 'small business

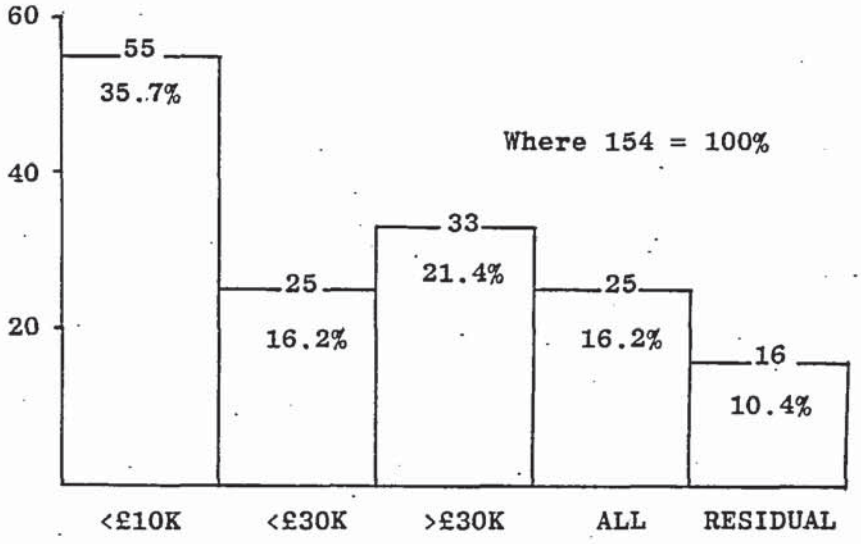
---

* Neither was computer hardware distinguished using terms such as 'microcomputer', 'minicomputer' and 'mainframe computer' in the questionnaire, for the demarcation between micro and mini, and mini and mainframe is not always clear.[4]

hardware systems'; category (c) comprises 'small business minicomputers'; and category (d) comprises 'supermini' and 'mainframe' computers. These terms are used hereafter to denote their respective hardware price ranges.[5]

One hundred and fifty-four replies were received, the distribution of answers being represented as a histogram in Figure 4.3 below.

Figure 4.3: Types of Hardware



35.7% of the respondents answering the question, representing 55 firms, ringed range (a) exclusively, and therefore the programs they supplied were primarily designed to run on microcomputers. Few of the remaining 99 respondents ringed one category exclusively. Twenty-five respondents, 16.2%, ringed either range (b) or ranges (a) and (b), and thus their firms' supplied programs to run on hardware costing less than £30,000, small business systems and microcomputers. A similar percentage of

respondents ringed all the price ranges. Thirty-three respondents, 21.4%, ringed range (c), range (d) or both, hardware costing more than £30,000. Their firms' programs were designed to run on small business minicomputers, superminis and mainframe computers. The remaining 10.4% ringed some ranges, but not others, and did not fall in any of the above sectors.

The term 'microcomputer software suppliers' as used hereafter, refer to those companies of the sample that supplied programs designed to run on hardware retailing at less than £10,000 per unit. Corresponding terms refer to the other types of firm in the sample.

Thus, although over a third of companies in the sample supplied microcomputer software exclusively, other groups of software suppliers were each well represented. Some of the most interesting results of the survey appeared when responses were classified according to the type of hardware. Indeed the survey revealed that the protection practices of most microcomputer software suppliers differed markedly from those of other sectors, and particularly mini and mainframe software suppliers in the sample.

INHERENT NATURE OF PROGRAMS

Most, if not all, programs require routine maintenance: (a) to fix errors, 'bugs', in the code; (b) to enhance the program, or suite of programs, by adding functions, 'modules', to it; (c) to amend the software following developments in computer

hardware; and (d) to keep pace with changing external conditions.

End users need for routine maintenance implies a 'post-sale' relationship between them and their suppliers. Contact is either direct, or indirect through intermediate dealers. Sooner or later all unauthorised end users would be forced to contact the, or a, bona fide software supplier in order to obtain vital updates of the program. The actual circumstances of his original purchase should then be discovered. Thus, in theory at least, maintenance services can provide protection to the supplier.

Respondents were asked whether end users' dependence on their firms for maintenance provided them with effective protection. They were asked to express their views on a scale of effectiveness divided into four divisions:

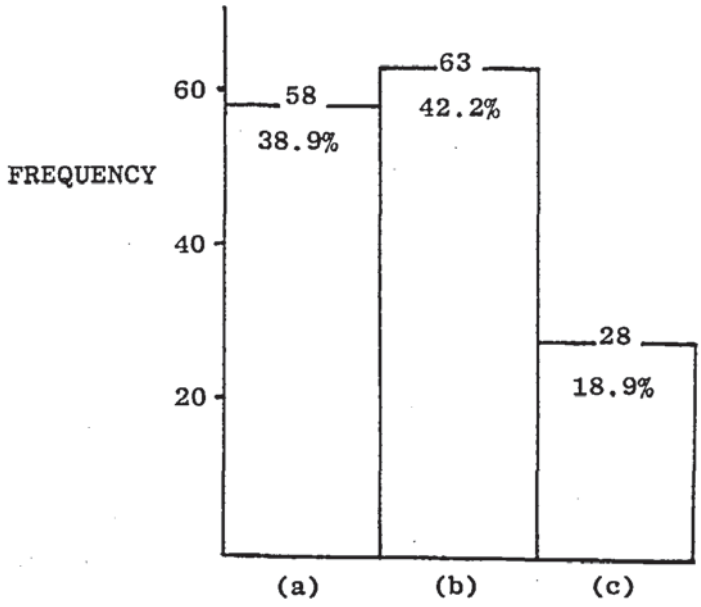
```
------------/------------/------------/------------
   ALWAYS        SOMETIMES     HARDLY EVER      NEVER
```

One hundred and forty-nine replies were received, and these opinions were distributed as shown in Figure 4.4.

The distribution of opinions indicate that there was no consensus in the sample on the effectiveness of protection provided by maintenance. Fifty-eight respondents (38.9% of opinions expressed) considered that, in their experience, it 'always' provided effective protection. However the largest minority, 63 respondents (42.2%), were more circumspect, and regarded it as merely 'sometimes' providing effective protection. The remaining 28 respondents venturing opinions (18.9%) considered that it provided nominal protection, at best 'hardly ever' being

effective.

Figure 4.4: Protection from Maintenance



Does the END USER'S dependence on your firm for maintenance confer effective protection?

Key: (a) ALWAYS
(b) SOMETIMES
(c) HARDLY EVER/ NEVER

Where 149 = 100%

The lack of a consensus was also discovered in each sector of the sample. Table 4.1. overleaf represents a breakdown of the overall distribution of opinions in Figure 4.4.

Fifty-two respondents from microcomputer software firms ventured opinions, and these were equally distributed in the three categories. However, the percentage of opinions residing in the 'hardly ever/never' cell (i.e., (c)) (32.7%) was substantially larger than those of respondents in other sectors expressing similar views for their software (ranging from 7.8% - 18.2%). Furthermore, the 17 respondents in this cell comprised over 60% of opinions expressed in the corresponding category of the overall

distribution. Thus, of those respondents who regarded maintenance as providing nominal protection, most were microcomputer software suppliers.

Table 4.1: Protection from Maintenance by Sector

TYPE OF FIRM	OPINIONS			NUMBER OF OPINIONS VENTURED PER SECTOR
	(a)	(b)	(c)	
Microcomputer Software Firms	19 (36.5%)	16 (30.8%)	17 (32.7%)	52 (100%)
Small Business Systems Software Firms	11 (45.8%)	11 (45.8%)	2 (8.3%)	24 (100%)
Mini, Supermini and Mainframe Software Firms	15 (45.5%)	12 (36.4%)	6 (18.2%)	33 (100%)
Remaining Software Firms	12 (31.6%)	23 (60.5%)	3 (7.8%)	38 (100%)

Some software suppliers each had a list of their authorised end users and would, as a matter of routine, check it upon each call for maintenance. If the end user was not on the list then he would not be authorised to use the software. Thus an end user's need for regular maintenance results in a protection procedure for the software supplier. However some respondents commented that they did not regard this procedure as foolproof.*

* 009, 099, 125.

Only two respondents thought maintenance provided their main protection.* Most respondents interviewed considered it a useful supplement to other forms of protection, but rarely relied upon it exclusively.

Most, if not all, programs are the result of an investment (sometimes a considerable investment) of time, money and expertise. In the course of software development new programming techniques may be developed. Alternatively, much time and money may be spent in assessing and rejecting alternative known techniques. In either case, the programmer, and where appropriate, his employer, would have built up a considerable body of 'know-how', from a familiarity with the problem, and the algorithm devised to solve it. This knowledge is usually confidential. Thus, in theory at least, the software supplier would have a lead time advantage over competitors. Effectively this is a temporary protection, for they (the competitors) would have to go through the same process of development in order to build up their own 'know-how'.

Respondents were asked whether the investment of time and money spent in developing the software provided effective protection. They were asked to express their opinions on a similar scale of effectiveness to the one mentioned previously, namely:

```
-------------/-------------/-------------/-------------
   ALWAYS        SOMETIMES     HARDLY EVER      NEVER
```

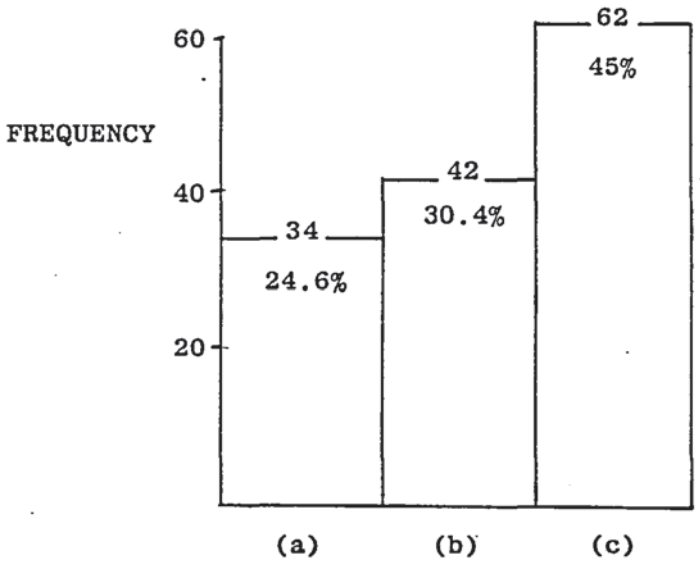
One hundred and thirty-eight respondents ventured opinions,

---

* 044, 098.

these being represented as a histogram in Figure 4.5.

Figure 4.5: Protection from Investment



Does the investment of time and money spent in developing the software confer effective protection?

Key: (a) ALWAYS
(b) SOMETIMES
(c) HARDLY EVER/ NEVER

Where 138 = 100%

Most of the respondents regarded investment as providing protection which was, at best, only 'sometimes' effective. Sixty-two respondents (representing 45% of opinions ventured) considered that, in their experience, investment provided nominal protection, 'hardly ever' being effective. Less than a quarter of opinions ventured were that it 'always' provided effective protection. Furthermore, of these, nobody commented that their firms relied upon it exclusively. Rather, a general consensus within the sample was that familiarity with the program was not a particularly effective form of protection.

This view was held, to a greater or lesser extent, by respondents in most sectors of the sample. Table 4.2 overleaf is a breakdown of the overall distribution of opinions in Figure 4.5.

Table 4.2: Protection from Investment by Sector

TYPE OF FIRM	O P I N I O N S			NUMBER OF OPINIONS VENTURED PER SECTOR
	(a)	(b)	(c)	
Microcomputer Software Firms	9 (18.8%)	11 (22.9%)	28 (58.4%)	48 (100%)
Small Business Systems Software Firms	4 (18.2%)	10 (45.4%)	8 (36.3%)	22 (100%)
Mini, Supermini and Mainframe Software Firms	12 (38.7%)	7 (22.6%)	12 (38.7%)	31 (100%)
Remaining Software Firms	8 (22.8%)	13 (37.1%)	14 (40%)	35 (100%)

Forty-eight respondents from microcomputer software firms ventured opinions. Twenty-eight of them (58.4%) considered that familiarity with the program, or suite of programs, provided hardly any effective protection (cell (c)). On the other hand, there was no consensus within the mini, supermini and mainframe software firms sector, for opinions expressed by 31 respondents therein were evenly distributed across the three categories.

A number of respondents commented that programs written for one, or a small number of, end users are inherently protected for the markets for these programs are small. Although this protection has not been considered, four respondents regarded the specialist nature of their programs as their main protection.*

* 062, 108, 143, 149.

EVIDENCE OF ABUSE OF RIGHTS

While many instances of unauthorised copying, or software piracy, have been reported in the trade press, the scale of the problem is unknown for such activities usually occur in private. In addition, the number of software suppliers that were/are suffering serious losses of software revenue from these activities is unknown.

Respondents were asked whether or not unauthorised copying of their programs, by end users, was commonplace. The word 'unauthorised' was used to distinguish between the legitimate needs of end users to make back-up copies of programs, from illicit copying, in which copies were destined for other end users without intending to pay the original supplier for them.

One hundred and fifty-five replies were received. Ninety-six respondents (61.9% of replies received) considered that, to their knowledge, their software was not being regularly copied by end users without authorisation. Of the remaining 59, only 24 respondents (15.4% of replies received) indicated that unauthorised copying by end users was commonplace.* The other 35 respondents wrote that they did not know. Thus for most software suppliers in the sample, unauthorised copying of their programs was rare. However, the 24 victim software firms should be interpreted as a lower bound, since there may be instances of

---

* 008, 009, 013, 023, 024, 027, 035, 036, 039, 051, 063, 077, 095, 096, 104, 107, 110, 115, 118, 126, 127, 133, 137, 147.

unauthorised copying which were not known by the relevant respondents when the survey was undertaken.

The distribution of these companies was not random throughout the sample, rather it was concentrated in particular sectors therein. In particular seventeen of these companies were microcomputer software suppliers.* Of the remaining 7 companies, six were small business systems software suppliers.# Thus 30.9% of microcomputer software firms in the sample knew that their programs had been regularly copied without authorisation, although they may not have known who had actually copied them. Furthermore, 16 of the 35 companies, whose respondents replied that they didn't know, were microcomputer software firms.

Although each unauthorised copy of a program, or suite of programs, constitutes a lost sale for the bona fide supplier, the twenty-four respondents from the victim companies were each asked whether such copying was regarded as a 'serious' loss of software revenue or whether it could readily be written off. Thirteen respondents considered that it did constitute a serious loss of business, of whom 11 represented microcomputer software suppliers.+

The results do not indicate the extent of unauthorised copying. Since it usually occurs in private, reliable data on the extent of such copying is probably unobtainable. Rather the

---

* 008, 009, 013, 023, 024, 027, 039, 051, 096, 104, 107, 110, 118, 126, 127, 133, 137.

\# 035, 036, 077, 095, 115, 147.

\+ 009, 013, 023, 024, 027, 039, 051, 096, 104, 107, 133.

results indicate the next best thing, that is the number and distribution of software suppliers in the sample that were aware that they have lost business from it. From this it appears that unauthorised copying by end users was the exception rather than the rule. However, it was (and maybe still is) a serious problem for 20% of the microcomputer software suppliers in the sample. This percentage, although too high, was nevertheless lower than expected. However, the microcomputer software suppliers in the sample did not, in the main, supply computer games and other personal microcomputer software. The problem may now be regarded as 'serious' by a higher proportion of microcomputer software suppliers in the industry, and particularly those developing and/or marketing software designed to run on home computers, as opposed to business computers.

Respondents were asked whether their firms were incurring serious losses in software revenue from piracy by dealers. Piracy was defined in the question as the unauthorised reproduction and sale for profit of the supplier's software.

One hundred and fifty-one replies were received. Ninety-seven firms in the sample had not, to the knowledge of their respondents, incurred losses in revenue from piracy. Of the remaining 54 replies, only 24 respondents (16%) replied that their firms suffered from it. In common with the previous results concerning unauthorised copying, these 24 firms were not randomly distributed throughout the sample. Rather 13 were microcomputer software suppliers* and a further 5 were small

---

* 009, 013, 023, 024, 027, 039, 047, 069, 096, 104, 107, 113, 133.

business systems software firms.* Thus 23% of microcomputer software suppliers in the sample knew that they were incurring losses from piracy. It is not surprising that microcomputer software firms have suffered more than other sectors because they rely more upon a dealer network for contact with potential customers. Indeed one respondent, representing a microcomputer software firm, commented that piracy was always a threat, but it was a risk that had to be taken for:

> "... Most of our dealers have no track record and we have to trust them or not sell."#

In common with the previous results concerning unauthorised copying, the data does not indicate the extent of software piracy. Rather it indicates the number of companies that were aware it was going on although they may not have known who was actually pirating their programs. Dishonest dealers rely upon the supplier's ignorance of these activities, and since it usually occurs in private, it is probable that the number of firms in the sample that have actually incurred 'serious' losses is larger than 24. However, the evidence is consistent with the interpretation that, in general, piracy is <u>not</u> endemic in the industry, at least for business, commercial and industrial software.

---

* 036, 040, 068, 095, 151.

\# 084.

LEGAL PROTECTION

The two main types of protection considered in the study were legal rights of the software supplier and any technical protection measures incorporated into programs. The results of questions on the former are reported below, and the results of questions concerning the latter, technical protection measures, are reported in the succeeding section.

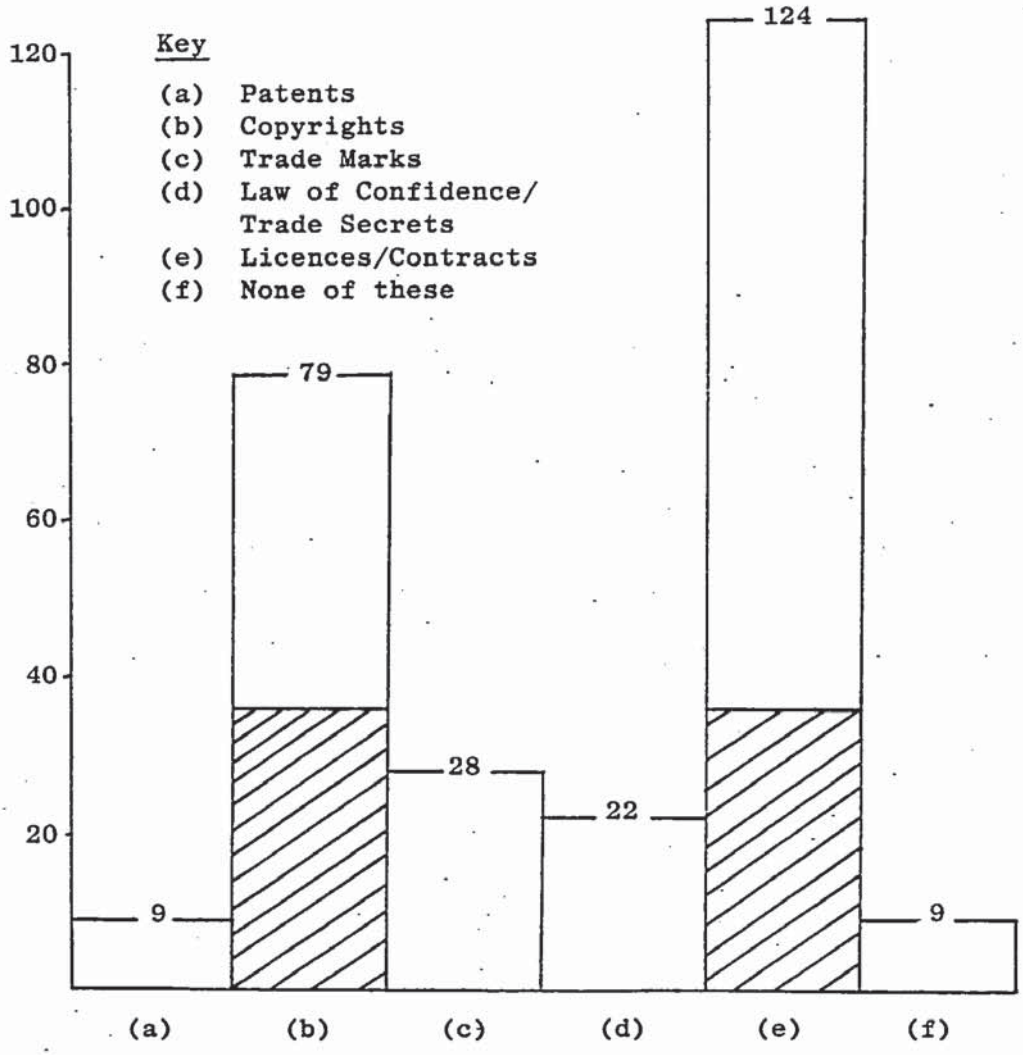
Respondents were asked to specify the legal rights used to protect their firms' software. One hundred and fifty-four replies were received and the distribution of answers is represented as a bar chart in Figure 4.6 overleaf.

By far the most popular mode of legal protection used was the licence or contract, specified by 124 respondents, representing over 80% of the software firms in the sample. Some respondents commented that their firms sold software outright to end users but required their dealers enter a contractual relationship with them (the suppliers). These firms are included in the 'licences/contracts' bar.

Forty-nine of these respondents specified licences/contracts exclusively. Other legal rights vesting with the supplier, (e.g. copyright in the software) were not, it seems, used. Eleven respondents from microcomputer software firms specified copyright exclusively. A further 9 respondents from microcomputer software firms specified various technical forms of protection exclusively. These comprise the 9 firms in the 'none of these' bar, (f), in Figure 4.6. Details of these measures are considered in the

next section. Of the remaining 83 replies, few respondents specified a single form of legal protection. Rather different combinations of legal rights were used. The most popular combination specified was copyright and licences, used by 36 software firms in the sample. These companies are included in both the relevant bars (b and e) of Figure 4.6 and are shaded.

Figure 4.6: Types of Legal Protection



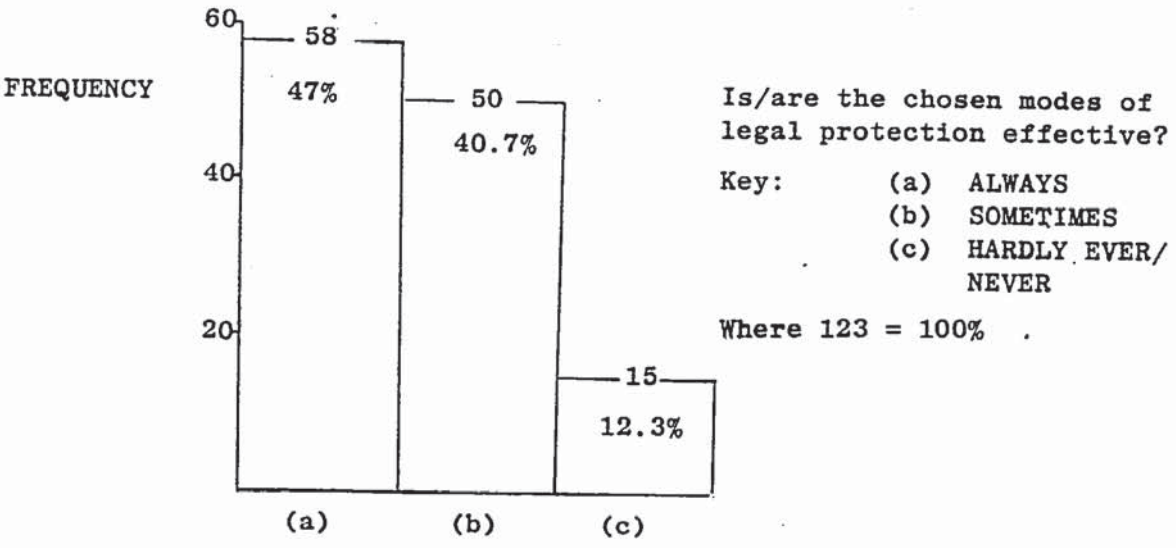
The next most popular legal right specified, inclusively or exclusively, was copyright, used by 79 companies, over 50% of software firms in the sample. The remaining forms of legal protection, that is, trade marks, confidentiality and patents, were less widely used. Only 9 software firms protected their software-inventions by patent.

Respondents were asked to venture an opinion on whether legal rights provided effective protection for their firms' software. They were asked to express their views on the following scale of effectiveness, similar to the ones used previously, namely:

```
-------------/-------------/-------------/-------------
   ALWAYS        SOMETIMES     HARDLY EVER       NEVER
```

One hundred and twenty-three respondents ventured an opinion, and the distribution of views obtained is represented as a histogram in Figure 4.7 below.

Figure 4.7: Opinions on Legal Protection



No overall consensus was discovered. However, in spite of the current legal uncertainties, the largest minority, 58 respondents (representing 47% of opinions expressed), considered that, in their experience, legal rights 'always' provided effective protection. Fifty respondents (representing 40.7% of opinions expressed) were more cautious and considered that legal rights were merely 'sometimes' effective. The remaining 15 respondents considered that, in their experience, legal rights provided, at best, nominal protection, and were hardly ever effective.

However, previous results indicated that most software firms in the sample have not knowingly suffered serious losses of business from unauthorised copying and/or piracy, therefore most have never had to enforce (or even threaten to enforce) their legal rights. Thus, these opinions were, for the most part, formed without first-hand experience, and thus must be regarded as opinions made in relative ignorance. Indeed 13 respondents wrote that they did not know how effective their legal rights were. Furthermore, one commented that since his firm had never had to enforce its rights, he had no evidence upon which to base an opinion. However, this does not apply to those 24 firms in the sample that had suffered serious losses of revenue from software piracy. These firms would almost certainly have sought professional legal advice, and, in the light of which, carefully considered their legal options. Twenty respondents from these firms ventured opinions. Only two considered that, from their experience, legal rights in their firms' software were 'always' effective. Thirteen considered that they were 'sometimes'

effective. The remaining 5 respondents considered that they provided 'hardly any' effective protection. Thus opinions of those respondents from firms that have had to enforce, or threaten to enforce, their legal rights were mixed. Most considered that the law was only partially effective.

A number of respondents commented that enforcing legal rights was too expensive and risky to be worthwhile.* Indeed one commented:

> "The present law of copyright is uncertain as regards to magnetic media. Anyone taking legal action in the High Court would almost certainly be faced with enormous legal bills with no certainty in the event of winning the action that the losing party would be able to pay costs awarded against it."#

While no overall consensus was found, marked differences of opinion were discovered between various sectors of the sample. Table 4.3 overleaf is a breakdown of the overall distribution of opinions in Figure 4.7.

Of 30 respondents from mini, supermini and mainframe software suppliers venturing opinions, 23 (67.3%) considered that the law was 'always' effective. Each of these companies used licences or contracts to protect their software. Thus there was general agreement that they (used either alone or in conjunction with other legal rights) provided effective protection. However, there was no consensus amongst the 34 respondents from

---

* 005, 009, 023, 027, 084.

\# 023.

Table 4.3: Opinions on Legal Protection by Sector

TYPE OF FIRM	OPINIONS			NUMBER OF OPINIONS* PER SECTOR	
	(a)	(b)	(c)	+	-
Microcomputer Software Firms	8 (23.5%)	17 (50%)	9 (26.5%)	34 (100%)	21
Small Business Systems Software Firms	6 (27.3%)	12 (54.5%)	4 (15.4%)	22 (100%)	2
Mini, Supermini and Mainframe Software Firms	23 (67.3%)	7 (23.3%)	0	30 (100%)	3
Remaining Software Firms	20 (57.1%)	13 (37.1%)	2 (5.8%)	35 (100%)	6

microcomputer software firms. Twenty-eight of these companies used licences/contracts and 21 respondents expressed an opinion. Only 6 regarded them as 'always' providing effective protection either alone or in conjunction with other legal rights. Thus the evidence is consistent with the interpretation that while licences/contracts are an effective form of protection for software firms supplying mini, supermini and mainframe software to a limited market, they become less effective for the protection of software firms supplying programs to a wider market, i.e., predominantly microcomputer software suppliers. Furthermore, of the 11 microcomputer software companies that used copyright exclusively, 9 respondents ventured opinions. Only two regarded it as 'always'

---

* + represents number of opinions expressed. - represents those respondents who expressed no view.

being effective. Most of the respondents not venturing opinions represented microcomputer software firms in the sample.

TECHNICAL PROTECTION

In none of the previous surveys of the Computing Services Industry (US or UK) has technical protection of programs been seriously considered. The conclusions of the CONTU survey in the US included a passing reference to the 'technological resourcefulness' of software firms as a means of protection, but the scope of this term was left substantially undefined.[6]

In the present survey, technical protection comprises any measures incorporated in the program, or suite of programs, designed to aid the supplier's control of its dissemination in the marketplace. The 'definition' includes measures designed to restrict the end user's ability to use the program (or suite of programs) on hardware other than an authorised computer, e.g., hardware-based peripheral protection devices. It includes measures designed to prevent end users, or dealers, from being able to copy the program or suite of programs. It also includes measures designed to deter end users, or dealers, from illicit copying or use of the program, as opposed to measures designed to prevent them from so doing.

The aims of questions in this section of the questionnaire were: (i) to determine the types of technical protection measures being used by software firms in the sample; (ii) the distribution of their use by sector; (iii) the importance of protection as a

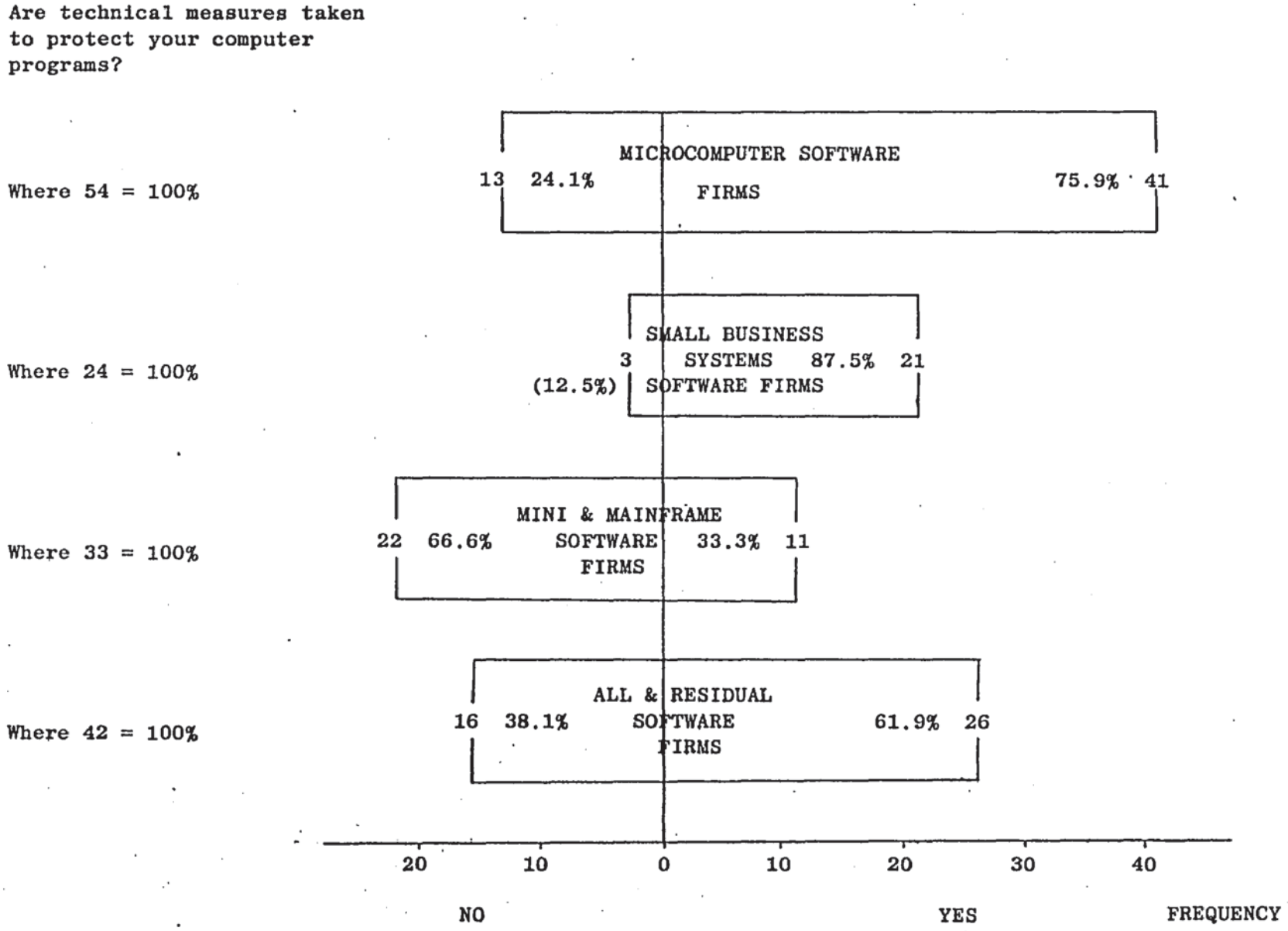
program design criterion; and (iv) the distribution of research then being conducted into technical protection.

Respondents were asked whether technical measures were taken to protect their firms' programs. The question was phrased to exclude measures designed to protect end users' data, and include only these measures designed to protect the supplier's programs. One hundred and fifty-three respondents replied. Ninety-nine (64.7%) stated that technical protection measures were taken to protect some or all of their firms' programs. The remaining 54 (35.3%) wrote on the questionnaire that measures were not taken, although when interviewed, a number commented that such measures had recently been introduced or that their policies were then under review. Thus, far from being unusual, it is common practice for software firms in the sample to take technical protection measures. However, the proportion of companies using such measures varied markedly from one sector to another within the sample. Figure 4.8 overleaf represents the distribution of replies to the above question broken down according to sector.

Of 54 replies received from microcomputer software firms, 41 respondents, representing 75.9% of replies received from this sector, stated that technical measures were used. Thus, most microcomputer software suppliers in the sample considered that there was a need for measures in addition to their legal rights.

In contrast, in the mini, supermini and mainframe sector of the sample, only 11 companies took such measures. The remaining 22 software firms (66.6%) did not consider it necessary,

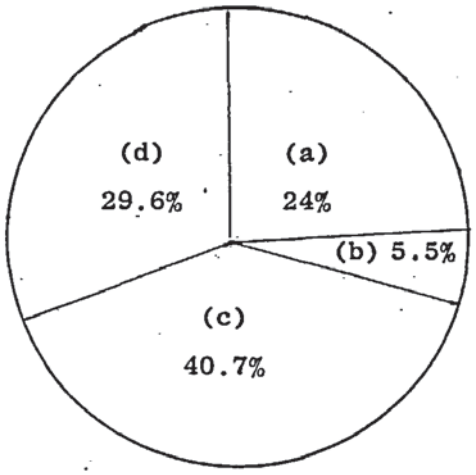
Figure 4.8: Use of Technical Protection by Sector



their legal rights being sufficient.

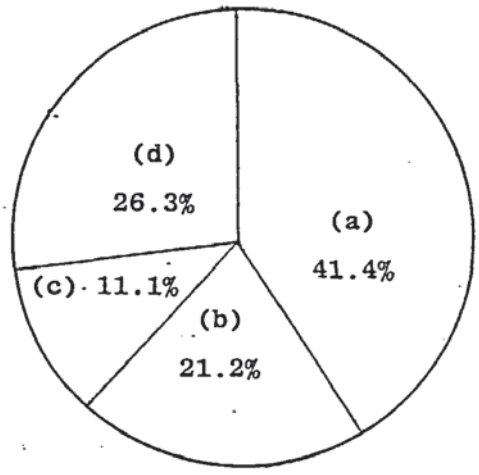
Figure 4.9 represents the distribution of companies within the sample taking, and not taking, technical protection measures. The pie chart on the right hand side, Figure 4.9(a), is a breakdown of the 99 companies taking measures; and the left hand pie chart, Figure 4.9(b), is a breakdown of the 54 companies that indicated on the questionnaire that such measures were not taken.

Figure 4.9: Distribution of Firms taking, and not taking, Technical Measures



Where 54 = 100%

Figure 4.9(b): Distribution of Firms not taking Technical Measures



Where 99 = 100%

Figure 4.9(a): Distribution of Firms taking Technical Measures

Key

(a) Microcomputer Software Firms
(b) Small Business Systems Software Firms
(c) Mini and Mainframe Software Firms
(d) All and Residual Software Firms

Microcomputer and small business systems software suppliers together comprised over 60% of those firms in the sample taking technical protection measures. Only 11.1% were mini and mainframe software suppliers.

Over 40% of software firms in the sample not taking technical protection measures were mini and mainframe software firms. This sector comprised only 21.4% of firms in the sample. Furthermore, microcomputer and small business software firms comprised less than 30% of the firms in this pie chart, yet the two sectors together comprised over half of the firms in the sample. Thus the evidence is consistent with the hypothesis that technical protection measures were mainly taken by those companies supplying programs designed to cater for the larger volume markets, that is, microcomputer software suppliers.

Types Of Technical Protection Measures Used

Given the obvious need to keep details of the measures confidential, respondents were asked to specify the criteria the measures were designed to satisfy, namely:

i) prevent/deter unauthorised copying of the programs?

(a) YES
(b) NO

If YES, please specify measures

.....

ii) prevent/deter unauthorised use of the programs?

(a) YES
(b) NO

If YES, please specify measures

.....

iii) aid discovery/proof of the source of the copy?

(a) YES
(b) NO

If YES, please specify measures

.....

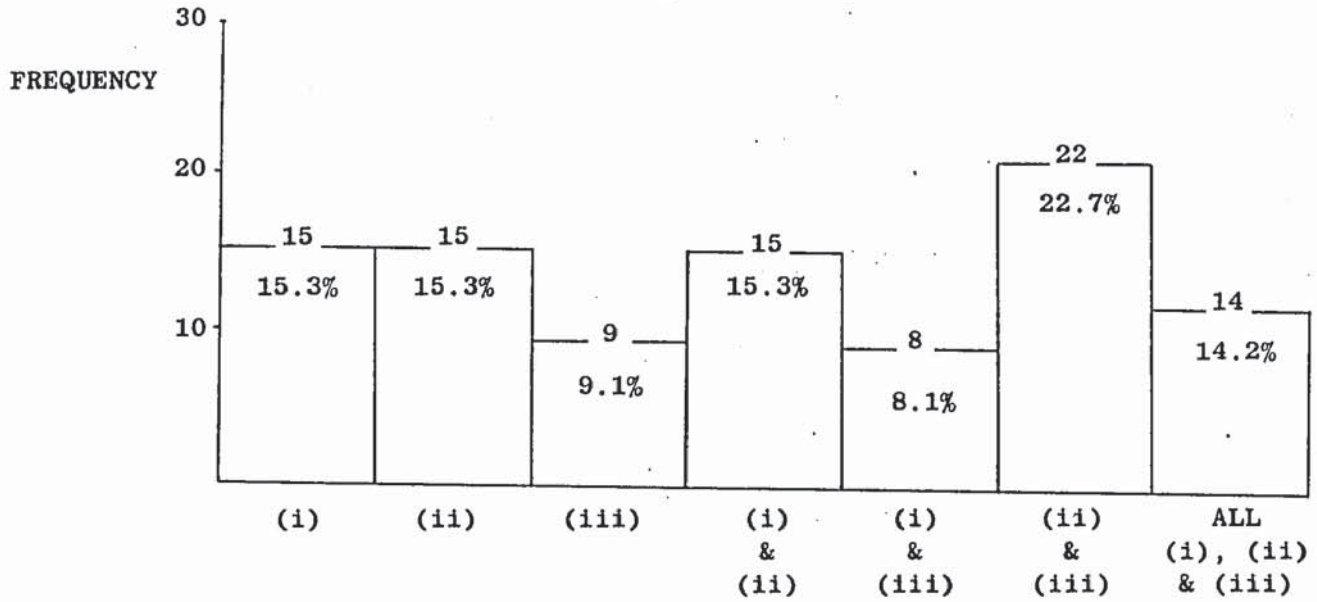
Ninety-eight respondents, from the 99 firms taking technical protection measures, answered at least the first part of questions (i) - (iii). The distribution of the YES replies are expressed as a bar chart in Figure 4.10 overleaf.

No single design criterion predominated. Indeed for 59 firms, the measures taken were designed to counter a number of different threats, for these respondents specified two or more design criteria. Each of the different combinations is included in Figure 4.10. Thus, 15 respondents specified that their measures were designed to prevent/deter unauthorised copying of their firms' programs. The same number specified that their measures were designed to prevent/deter unauthorised use; and so on.

Respondents were invited to specify, in greater detail, the measures taken. Eighty-one did so, either on the questionnaire itself or in follow-up interviews.

Each of these companies supplied programs in machine code or intermediate compiled code. Source code was hardly ever released

Figure 4.10: Types of Technical Measures Taken



Key

(i) Means designed to prevent/deter unauthorised copying

(ii) Means designed to prevent/deter unauthorised use

(iii) Means designed to aid proof of source of copy

Where 98 = 100%

to end users.

Two firms supplied programs in their own specialist computer languages, rather than rely upon one of the more relatively well known, and from their point of view, less secure languages.*

Fourteen firms used hardware-based protection devices. Most of these devices had to be connected externally to the chassis of the particular microcomputer at the relevant port.# One firm used a protection device which, in operation, had to be inserted into the chassis of a particular type of microcomputer. Each 'add-on', or 'add-in', device was designed to allow the end user to copy the program, but not use it in data processing without it. In all cases its presence was required at the relevant port, or memory slot, to allow data processing to proceed. Each of the programs was designed to stop processing, once, or soon after, it had detected removal of the device or tampering of it. At least 6 of these firms manufactured their own devices.+

Sixteen firms used copy-protect means designed to prevent unauthorised copying of their programs.≠ Most of these firms devised their own measures. The techniques used were various, including modifying the format of the disks, the media containing

---

* 006, 120.

\# 002, 015, 025, 030, 038, 059, 077, 089, 096, 103, 121, 125, 139, 144.

\+ 002, 089, 096, 103, 125, 147.

≠ 009, 012, 013, 021, 023, 029, 034, 035, 051, 053, 054, 059, 068, 073, 118, 153.

the programs, so that when the operating system copy command was invoked it could not identify the protected sectors or tracks, and therefore could not copy them.

The most widely specified method of protection was the serialisation, or personalisation, of programs. In each case a particular copy would be tied to a particular dealer or end user. Twenty-eight of the 81 firms used this type of protection.* The measures were psychological in that they were not designed to prevent unauthorised copying or use, rather deter such activities. Although there were many variations on a theme, one common technique was to insert an encrypted form of the authorised end user's name and address at various points throughout a suite of programs. A plain text version of this data would always be displayed while the suite was in use, so that there would be no doubt as to its rightful owner. This was sometimes, although not always it seems, combined with a copyright notice.

Eight firms used date, or count, locks so that after a certain date, or number of runs, the end user would be forced to contact the supplier for continued use of the software.#

Some respondents specified combinations of the above.

Other types of measures included tying the applications programs to the operating system programs by inserting corresponding

---

* 004, 024, 036, 042, 046, 050, 052, 057, 058, 066, 080, 083, 084, 093, 095, 097, 099, 100, 106, 109, 124, 130, 131, 133, 140, 146, 150, 156.

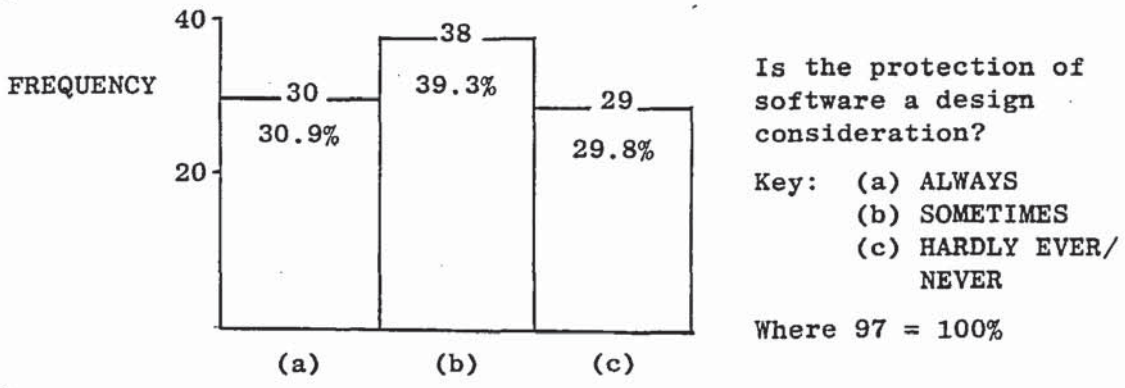
\# 018, 055, 063, 075, 095, 109, 111, 117.

serial numbers into both. Thus a particular copy would be tied to a particular copy of the operating system, without which it would not run. One firm tied its software to the end user's particular hardware configuration so that it would not run on any other configuration.

Most of the respondents interviewed considered that a determined 'pirate' with sufficient time and skill would sooner or later by-pass, or 'crack', their technical measures. Many commented that deterring amateur, or casual, copying was all that could realistically be achieved and to this extent they regarded their measures as always being effective.

Respondents were asked whether protection was a design criterion in the development of their firms' programs. They were asked to express their views on a scale of importance similar to the ones used previously. Ninety-seven respondents from firms taking technical protection measures replied, and the distribution of their answers are shown in Figure 4.11 below.

Figure 4.11: Protection as a Design Consideration



Given the variety of technical measures specified, it was perhaps not surprising that the importance of protection varied from one firm to another in the sample. Thus while 30 respondents considered that protection was 'always' a consideration in software design, 29 considered that it was 'hardly ever' a consideration. Furthermore, within each sector of the sample there was, likewise, no consensus.

This was surprising, since it was expected that most of the firms in the microcomputer software and small business systems software sectors, the main developers and users of technical protection, would consider it a significant design consideration.

In general the preferred mode of protection (legal or technical) depends upon many factors including the size of the market for the program, the threat of unauthorised copying or piracy, its retail price, and its popularity with end users. The relative importance of some of these factors is not apparent until after the program has first been marketed, for only then can its popularity with end users be gauged. Thus technical protection may not initially be a consideration in design, but subsequently become one. There seems to be a protection learning curve, and from interviews with respondents, the sample comprised firms at each of its points. Some were at the stage of first marketing and had not formulated any, but the most rudimentary of, protection policies. Others had marketed their programs over many months, and had incorporated sophisticated technical protection measures into updates or enhancements of

their popular programs.

The distribution reflects the many different types of technical measures taken. Some were bought in, or were relatively incidental, and in each case required little modification to the program. Others, on the other hand, required substantial modifications to programs, and thus were 'always' a design consideration.

Research into Technical Protection

Respondents were asked whether their firms' were allocating resources to protection, and if so, to specify the form this allocation took. One hundred and fifty-two replies were received, of which 48 respondents* (31.5% of replies received) wrote that their firms were then allocating resources to protection. Of these, 38 respondents specified the form the allocation took, either on the questionnaire itself or in follow-up interviews. The remaining 10 regarded this information as confidential, although all took technical measures.

Three respondents wrote that they were in the process of taking legal proceedings and that their allocation of resources were solicitors fees.# Each of the remaining 35 were allocating resources to the technical protection of their programs. For 12 of these firms, in each case, protection was the responsibility

---

* 002, 006, 008, 009, 015, 018, 025, 032, 035, 036, 037, 038, 047, 048, 051, 059, 068, 079, 083, 084, 085, 087, 098, 093, 096, 100, 103, 107, 115, 116, 118, 121, 124, 125, 126, 127, 130, 131, 133, 134, 135, 137, 138, 146, 150, 153, 156.

\# 008, 037, 079.

of the software development manager, and the allocation of resources took the form of a longer development time for their respective programs.* Six companies were conducting research into hardware-based protection devices.# For one of these companies their stake in this field amounted to an investment of £100,000 per annum.+ A further 6 companies bought in software protection products from another, or other suppliers,≠ and four firms employed encryption experts on a part-time or full-time basis.°

These firms were not randomly distributed in the sample, rather they were concentrated in particular sectors therein. Figure 4.12 overleaf is a breakdown of the 45 companies in the sample which were then allocating resources to the technical protection of their programs.≠

Three-quarters of the companies that were then conducting research into technical protection of programs or buying-in such protection were either microcomputer or small business systems software suppliers. None of the mini, supermini and mainframe software firms in the sample were allocating resources to the protection of their software. Thus interest in technical

---

* 009, 018, 036, 047, 068, 083, 084, 085, 093, 100, 115, 150.

\# 002, 089, 096, 103, 125, 147.

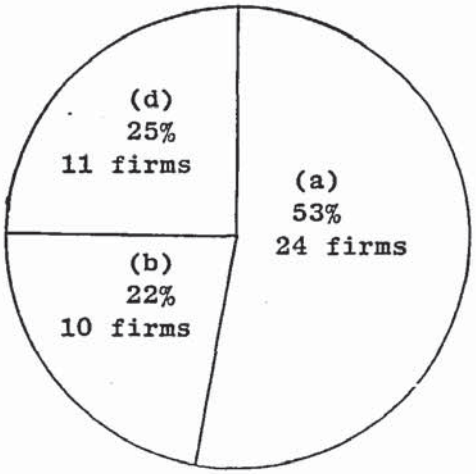
\+ 096.

≠ 015, 025, 051, 059, 121, 130.

° 118, 121, 124, 127.

≠ It is assumed that the 10 companies that declined to elaborate were allocating resources to the technical protection of their programs.

Figure 4.12: Allocation of Resources into Technical Protection



Where 45 = 100%

Key

(a) Microcomputer Software Firms
(b) Small Business Systems Software Firms
(c) Mini et al. Software Firms
(d) All and Residual Software firms

protection was mainly confined to those companies supplying software to the larger volume markets.

CONCLUSIONS - PREFERRED MODES OF PROTECTION

Few respondents considered that end user's need for regular maintenance of programs provided sufficient nor effective protection. Some considered that, in their experience, it had always proved to be effective, but few relied upon it exclusively. Rather, for most companies in the sample, it sometimes provided a useful supplement to other forms of protection. Familiarity with the program, or suite of programs, provided little inherent protection in the opinion of most respondents to the questionnaire. Thus inherent features of programs were hardly ever relied upon exclusively, although a small number of respondents commented that the specialist nature of their programs provided their firms' main protection. For most companies in the sample their main

protection resided elsewhere, and particularly in their legal rights in, and the technical protection features of, their software.

Referring to legal protection, while licences, contracts, and to a lesser extent copyright, were widely used, there was no overall consensus on their effectiveness. This may be due to the lack of experience in the enforcement of legal rights for only 36 companies in the sample had knowingly lost business from either unauthorised private copying and/or piracy of their programs. However, of those that had suffered, only a small handful of respondents considered that, in the light of their firms' experiences, the law was always effective. Rather some commented that licences were an inappropriate form of protection for their software, whereas others commented on the practical problems of the enforcement of rights in court. The following comment made by one respondent summarised most the others made:

> "The methods of legal protection [Patents, Copyrights, Trade Marks, and Law of Confidence] involve expense and delay and are not practicable in my opinion. Licences are a possibility but when selling through dealers still offer little protection ..."*

Within some sectors of the sample, however, a consensus was discovered, and particularly respondents representing mini, super-mini and mainframe software suppliers. Most considered that, in their experience, licences or contracts always provided effective

---

* 084.

protection. While there was no consensus amongst respondents from microcomputer software firms in the sample, only 8 considered that licenses/contracts and/or copyright were always effective. Furthermore, seven of these companies also used technical protection measures. Thus, in spite of this expression of confidence in the law, these companies did not wholly rely upon it. Rather they used a combination of legal and technical protection methods. Twenty-six of the remaining 47 respondents in this sector considered licensing and copyright to be only partially effective in the protection of their firms' software. The other twenty-one respondents did not venture any opinions.

Far from being unusual, it was common practice for firms in the sample to take technical protection measures. The measures taken were various, and included hardware-based peripheral devices designed to prevent unauthorised use of the protected program, and software designed to restrict copying of the program. The most popular type of protection measures specified was those designed to deter, rather than prevent, unauthorised copying or use.

Firms taking technical protection measures were mainly microcomputer and small business systems software suppliers. The majority of mini, supermini and mainframe software firms did not use such measures. Furthermore, when the survey was undertaken, most of the companies buying-in technical protection devices or programs, or conducting research into methods of protection were companies in the former category.

Thus, interest in technical protection was, for the most part, confined to microcomputer and small business systems software suppliers. The evidence is consistent with the interpretation that a lack of confidence in the law (in its applicability and/or its enforcement) has resulted in the development and use of technical protection measures.

Respondents were asked whether or not a lack of legal protection had influenced innovations in computing. Clearly the development and use of technical protection measures are examples of innovations resulting from a perceived lack of effective legal, and inherent, protection. Of 142 replies received, 58 respondents agreed with the hypothesis (40.8%), and 84 (59.2%) disagreed. However of 49 replies from respondents representing microcomputer software suppliers, 26 (53.1%) agreed. Most of these companies took technical protection measures.* Furthermore, most of the respondents had previously expressed the view that their firms' legal rights were only partially effective.#

The remaining 23 respondents disagreed (46.9%). Some mentioned other consequences of inadequate legal protection. Amongst other comments made, some considered that there could be restrictions on the availability of cheap software for large volume markets,+ whilst others thought that many small programs remained

---

* 004, 009, 013, 023, 061, 078, 085, 087, 096, 105, 113, 118, 122, 133, 134, 137, 146, 153, 156.

\# 003, 009, 023, 027, 061, 069, 071, 087, 096, 104, 113, 118, 133, 134, 137, 146, 153, 156.

\+ 036.

unpublished from the threat of piracy.* A number commented that the threat of piracy could deter investment in new applications.#

However, of these 23 companies, 20 were taking technical protection measures+ and 11 were also allocating resources to protection when the survey was undertaken.≠ Thus, in spite of this large dissenting minority, in practice many of their firms' policies stemmed from a lack of confidence in the law. However, the technical protection measures resulting, were each regarded as an incidental program design criterion. In the opinion of these respondents, such measures did not constitute significant innovations in computing.

In contrast, of 31 opinions expressed by respondents from mini, supermini and mainframe software firms, 25 disagreed with the hypothesis.° Most had previously indicated that, in their view, licensing and copyright always provided effective protection,≠ and that together with inherent protection, they provided sufficient protection for their firms' programs.

Thus, within the industry, different sectors protect their software in different ways. For many mini, supermini and

---

* 042.

\# 083, 120.

\+ 006, 012, 014, 015, 024, 030, 034, 047, 050, 051, 059, 066, 075, 089, 107, 125, 126, 127, 130, 139.

≠ 006, 015, 047, 051, 059, 089, 107, 125, 126, 127, 130.

° 016, 019, 022, 028, 029, 031, 043, 044, 046, 053, 057, 060, 082, 088, 098, 109, 115, 116, 119, 123, 132, 136, 138, 140, 145.

≠ 016, 019, 022, 031, 042, 053, 057, 060, 088, 098, 116, 119, 123, 132, 136, 138, 145.

mainframe software firms, together with some companies in other sectors of the sample, inherent features of their programs, and their legal rights, provide effective protection. However, for most microcomputer and small business systems software suppliers in the sample, inherent features, licensing and copyright are considered to be only partially effective, and therefore reliance is placed more upon technical protection measures incorporated into their programs.

CHAPTER 5

TECHNOLOGICAL PROTECTION OF COMPUTER PROGRAMS

INTRODUCTION

In the last few years, the widespread availability of cheap microcomputers has resulted in a mass market for compatible programs.[1] It has also resulted in a marked change in the location of data processing from centralised computer bureaux and data processing departments, to decentralised microcomputers.[2] While the non-exclusive licence provided effective protection for the computer software supplier when his market comprised a limited number of end users, controlling the dissemination of his programs, or suites of programs, became a problem when licensing identical copies to a market comprising a large number of microcomputer users. Each unauthorised user represented a loss of software revenue to the supplier. In addition, given the large number of authorised end users, and dealers, any of whom could have supplied the program, or suite of programs, tracing the source of the unauthorised copy became a major detection problem for the supplier. He knew that his licence had been infringed but he did not know by whom. Furthermore, even if he was successful and identified the infringer(s), the uncertain scope of his intellectual property rights, together with the crippling costs of their enforcement in court, meant that, for many of the smaller suppliers, litigation was an option that was simply not feasible. As a result many microcomputer software suppliers supplemented their legal rights

with technical protection measures incorporated into their programs.

This chapter describes some of the techniques that are being applied to protect computer programs from unauthorised copying and use.

While the survey, reported previously, discovered that the majority of microcomputer software suppliers (over 75%) took technical protection measures, details were rarely published. Furthermore, to date, little has been published generally on the technology of software protection.

This should be distinguished from literature describing techniques to protect data, on which, in comparison, much has already been published. Some of these techniques are useful in certain software protection schemes,* but, in general, they are designed for a different purpose. Data protection schemes are designed to prevent unauthorised access to information[3] expressed in a form in which it can be understood by an unauthorised person. In contrast, software protection schemes are designed to control use of the program, or suite of programs. There is no requirement that an unauthorised user should understand the structure, or operation of the software. He merely wants to be able to use it in processing his, or others', data. It follows that proven techniques in protecting data may not, on their own be effective

---

* A software protection scheme comprises one or more technical protection measures, incorporated in a program, or suite of programs, for its protection. Each measure, or feature counters a particular threat to the software.

in protecting software. A data protection scheme which prevents disclosure of the program in an understandable form, but nevertheless permits unimpeded use of it in a computer, is not an effective software protection scheme. Nevertheless, certain data protection techniques are invaluable in countering particular threats. For example, use of the Data Encryption Standard,* or public key encryption,# to encrypt communications between the Central Processing Unit (CPU) of a microcomputer and an interfacing peripheral device, along an otherwise unprotected bus, promises to diminish the threat posed by traffic analysis of these communications. Data protection techniques play a minor role in the software protection features described below.

One of the few published studies on methods of protecting software is a doctoral thesis entitled "Protecting externally supplied software in small computers" by S.T. Kent.[6] He describes two alternative protection methods, an 'encrypted bus approach' and an 'encrypted storage approach'.[7] In the encrypted bus approach, the microcomputer system is divided into

---

* The Data Encryption Standard is a block cipher developed by IBM. It comprises a 56-bit key which encrypts and decrypts blocks of data 64 bits in length. The encryption and decryption processes are each divided into rounds. Each round involves dividing the 64-bit block into two 32-bit half blocks; and each half block is then permutated and substituted with the other half block.[4]

\# The public key encryption technique has two keys, a public key and a private key known only to the user. The two keys are mathematically related. The public key is used to encrypt data, and the private key is used to decrypt the ciphertext.[5]

a number of parts, each part being encapsulated in a 'tamper resistant module' (TRM). Each TRM is designed so that data within it cannot be disclosed, or modified, by an intruder without also destroying the data.[8] Communication between the TRM parts is provided by a physically unprotected bus. The information transmitted on this bus is protected by cryptography. Six rounds of the Data Encryption Standard (DES) are used to encrypt the information, prior to its transmission from the TRM.[9] In the encrypted storage approach, the microprocessor and some memory are encapsulated in a single TRM. The remaining storage elements are physically unprotected. The DES is used to encrypt data stored in these elements and in communications with other elements of the system. The data is decrypted inside the TRM. Should it be necessary to subsequently store the processed data outside the TRM, it is encrypted prior to transmission to the unprotected element.[10]

Although tamper resistant modules have been used, neither of Kent's methods has yet been implemented in full.

The protection of programs is rarely a design consideration in the development of a microcomputer system.* This stems primarily from the structure of the microcomputer industry. As a first approximation, computer companies in the industry can be

---

* A microcomputer system as defined herein means the microcomputer hardware and operating systems software required to implement the program.

divided into two groups, microcomputer hardware manufacturers and independent software suppliers.[11] The hardware manufacturers are primarily in the business of manufacturing, and marketing, microcomputer hardware. In general, a minor portion of their revenue stems from software.[12] However, given their established network of hardware dealers, some market independents' programs provided, of course, that they are compatible with their hardware.* Furthermore, some appreciate the need for technical protection of programs. Indeed, one hardware manufacturer has devised, and supplies to approved dealers, a software protection package. This suite of programs is designed to modify other programs so that end users cannot copy the latter using particular operating systems software.[13]

The hardware architecture of some microcomputers inherently aids the protection of independents' programs. For example, the architecture of a range of microcomputers of a particular manufacturer permits the interfacing of 'intelligent' peripheral equipment# to the CPU.[14] This allows the application of scrambling and other data protection techniques in certain software protection schemes involving communication between the CPU and a peripheral software protection device. However, in general, the current architecture of popular microcomputer hardware and operating

---

* For example, Commodores' Approved Products Catalogue and Atari's Program Exchange (APEX).

\# Each peripheral device includes one or more microprocessors, in effect a microcomputer in its own right. The microcomputer system comprises a host microcomputer and intelligent peripheral devices which can also process data.

systems software offer little inherent protection. Thus, should an independent software supplier require technical protection, then he must either devise it himself, or buy-in protection from another supplier. Yet, each software protection scheme is implemented on the relevant 'open' hardware and operating systems software.

Some programs are inherently protected. All bespoke programs are, by definition tailored to one, or a small number of, end users. It is highly unlikely that there would be a wider market for such products. Thus, additional technical protection measures would not normally be required.

Mass market programs are rarely supplied to end users in source code form. Depending on the high level language, many are supplied in compiled, or assembled, form. The lack of source code makes it difficult, even for a knowledgeable programmer, to determine its structure, for compiled code contains no explanatory labels required to guide him through it.[15] Furthermore, compiled programs cannot be listed, for they are not in a form which the appropriate operating systems software can carry out LIST commands. Thus each such program cannot readily be analysed by an end user. However, many microcomputer software suppliers do not regard this protection as sufficient.

All technical protection measures are incorporated into the supplier's program, or suite of programs as the case may be,* and

---

* All references to a program include references to a suite of programs in a single software package, unless otherwise indicated.

are designed to aid his control of its dissemination in the market. This definition implies that the supplier cannot rely upon the end user, or intermediate dealer, to implement the software protection scheme. From the supplier's point of view, each is a threat to his future software revenue. Thus most password schemes are inherently unsafe, for the supplier has to trust the end user to keep the password confidential.

Given the obvious threat posed by end users, intermediate dealers and competitors, details of protection measures are invariably regarded as company confidential. As a result, many software protection schemes are devised in-house. Although many companies have devised similar measures, there is no universally-adopted software protection scheme. Rather, different schemes have resulted. In each case, the scheme is either tailored to the individual supplier's requirements, or it depends on what he can afford.

Some firms specialise in supplying software protection products, their markets being software suppliers who perceive a need for technical protection of their programs but neither have sufficient resources, nor expertise in-house, to devise suitable protection techniques themselves. Thus, a number of software houses use similar protection schemes because they have bought-in protection from one supplier.

However, on the basis of 31 visits and the survey of 156 software firms reported previously, current software protection schemes can be categorised as follows:

(a) schemes designed to effectively prevent unauthorised use of microcomputer programs;

(b) schemes designed to effectively prevent unauthorised copying of microcomputer programs; and

(c) schemes designed to deter unauthorised copying and use of microcomputer programs.

Each of these schemes contains various protection features. Each feature is designed to counter a particular threat. These features are not necessarily confined to schemes in one category, for certain features are applied in schemes residing in different categories. Furthermore, at least one software supplier uses two schemes residing in different categories, in a hybrid software protection scheme.[16]

USE-PROTECT MEASURES

Many software protection schemes designed to restrict use of the program, allow unrestricted copying of it. This is an important advantage over alternative schemes. Floppy disks, the predominant medium for supplying software to end users, tend to wear out with regular use. All of the alternative copy-protect schemes restrict the authorised end user's ability to copy the protected program. This is an inconvenience for the end user, for he is now dependent upon the supplier for back-up copies. It is also inconvenient for the supplier, for he has additional administrative problems inherent in an undertaking to supply such copies promptly. Copy-protect measures are described in greater detail in the succeeding section.

Each of the measures described in this section is included

in an implemented software protection scheme designed to permit 'authorised' use of the program. The software supplier distinguishes what he regards as authorised use, from unauthorised use, and incorporates this distinction into his software protection scheme. The measures described below are each designed to identify authorised from unauthorised use. Each such software protection scheme contains a 'stop and catch fire' routine,[17] an instruction to stop data processing and lock out the operator should use be deemed to be unauthorised.

Software-based Measures

One type of software-based use-protect measure is a date, or time, lock. In a typical scheme, access to the program depends upon the date. The operator is required to type in today's date on his keyboard, and this is compared with the expiry date written into the program. Should the operator attempt to use it after the expiry date, this is deemed to be an attempt to use it without authorisation, and it branches to a subroutine locking out his access to it.[18] Continued use depends on the supplier modifying the program, and particularly his insertion of a new expiry date once the user has paid him the required royalty.

A second category of software-based schemes is designed to 'tie' the program to a particular microcomputer system. This implies that every copy of it must be able to distinguish 'authorised' microcomputer systems from 'unauthorised' ones, under program control. The software protection scheme

incorporated within the program is designed to prevent use of it (or parts therein, for a suite of programs) on unauthorised systems. However, in many hardware manufacturers' product ranges, individual microcomputers are identical. There is rarely a feature within the hardware which can be used to tie a copy of a program to one, or a small number of, 'authorised' microcomputers.

One approach is to tie the program to the operating systems software, rather than to the hardware. In one such scheme both it and the systems software contain a serial number distinguishing each from every other copy. The serial number of the systems software is compared with the corresponding one in the program.[19] If it contains the 'correct' serial number, processing is deemed to be permitted. If not, then processing is deemed to be unauthorised, and it branches to a routine within the program designed to both stop further processing, and lock out the operator from further access to the program. This scheme is effective if, and only if, the serial number within the operating systems software is unique. If this software can be copied then clearly the protection is compromised for the serial number is no longer unique. Some suppliers have rejected a serial number, or other distinguishing feature, expressed in software, and instead opted for features incorporated in a 'firmware' or hardware-based device.

Hardware-based Measures

In many of these software protection schemes, each end user is required to attach a hardware device to a specific interface

of his microcomputer in order to use the relevant program in data processing. This peripheral device contains a feature differentiating the end user's microcomputer from other similar ones. Alternatively, one software supplier inserts a hardware device into the chassis of each of its end user's microcomputers.[20] It has the same function as the previous peripheral devices.

A simple protection device used by one software supplier is a Read Only Memory (ROM), having a pin configuration so that it could be attached to a particular interface of the Commodore 3032 microcomputer. The program simply checked its presence at the interface at selected points in processing.[21] Provided a suitable response was obtained from it, processing was allowed to proceed. A more sophisticated device is a Programmable Read Only Memory (PROM) containing a unique serial number. The program could then distinguish between similar PROMs. It would not run unless the correct PROM was located at the relevant interface.

Both of these methods provided effective protection until it became possible to manufacture Erasable Programmable Read Only Memories (EPROMs), cheaply. In the last few years the widespread availability of both EPROMs and the necessary copying equipment, made this protection redundant. Although the existence of such copiers may not be known to many first-time end users, they are well-known within the industry. Most PROMs and ROMs have EPROM equivalents, that is EPROMs having the same configuration, and thickness, of pins as these 'chips'. Therefore, the EPROM copies could replace a supplier's devices. The program would

now run on 'unauthorised' microcomputers. EPROM copiers defeated the protection, for it was possible not only to copy the program using standard copy utilities of the relevant operating system, but also its protection, the ROM or PROM distinguishing the microcomputer system from other similar ones.[22]

To diminish the threat of an intruder being able to physically copy the peripheral device, some microcomputer software suppliers encapsulated their devices and associated circuitry in epoxy resin. The intruder must first remove the resin to get at the integrated circuits inside to discover the serial number or other distinguishing feature. Some of these devices are designed so that attempts to remove the resin using certain techniques also destroy the circuits inside. A well-known TRM of this type is called a 'DONGLE'.* This device was first used to protect a wordprocessing suite of programs called 'Wordcraft' designed to run on the Commodore PET microcomputer. A number of other companies are using similar TRMs to protect their own programs. TRM protection is not confined to Commodore hardware for a number of companies supplying programs designed to run on Sirius[23] and Apple[24] hardware have each applied a similar technique. However, the individual software protection schemes differ from one company to another within this TRM family.

---

* The term 'Dongle' as defined by Paul Handover, Managing Director of Dataview Ltd., and reported on p. 8, of the 5th March 1981 edition of Computer Weekly, was as follows:

> "A dongle is a small device attached to the hardware which interfaces uniquely with each authorised copy of the software, and without which the package will not function."

Some software suppliers encase the epoxy resin in seamless metal, or plastic, containers to make it even more difficult for an intruder to get at the circuits inside.[25] Furthermore, in one supplier's TRMs, tiny glass beads are impregnated into the resin. Should an intruder attempt to remove it by slicing it along one of its strain lines, the glass beads would sever adjacent lines. Instead of a clean cut along one line revealing the circuits intact, the net result would be a crumpling of the device and the circuits inside would be shattered. In addition, the resin used by this supplier becomes highly conductive when heated. Should an intruder melt it in an attempt to separate it from the circuits, the connections therein are destroyed by the heating process.[26] Another software supplier changed the labels on the integrated circuit chips contained in his company's Dongles.[27] Should an intruder succeed in discovering the layout of the circuits in the device, and fabricate a similar one himself, the resulting device would not only not work, but also corrupt the software.

Measures designed to Foil some Software-based Intruder Techniques

While the above measures minimise the danger of physically copying the device, an intruder could, alternatively, attempt to by-pass the TRM by modifying the program. The program is usually stored in a unprotected memory, allowing him, effectively, unrestricted access to it. The intruder could use the operating systems software to isolate the TRM from the remainder of the program by 'patching', or by-passing, all interrogation

instructions to the TRM, and modifying subsequent COMPARE and other relevant instructions of the software protection routine in the program. A successful software attack of this type requires locating and identifying the relevant instructions. Some of the techniques used to prevent such an attack from succeeding, such as restricting the end user's access to the program, are described in the succeeding section, for they are also required in copy-protect schemes.

One of the techniques used to minimise this threat is to disguise these instructions, so that they are not readily identifiable to an intruder. One microcomputer software supplier encodes the interrogation instructions to the input/output port where the TRM is located. The instructions are only decoded immediately prior to transmission and are re-encoded soon after. From an intruder's point of view, a listing of the encoded instruction looks like data, unless he is fortunate enough to interrupt data processing after the instruction has been decoded, but before it has been re-encoded.[28] The interrogation instructions can also be disguised if they perform some other role in data processing, independent of the TRM. Thus should an intruder discover, and identify, an interrogation instruction to the TRM in the program and patch over it, processing is disrupted because he has also by-passed another important instruction.[29]

A second approach used by many software suppliers to minimise the probability of a successful software attack, is to branch to the software protection routine at various points throughout the execution of the program. Should an intruder

succeed in identifying, and by-passing, one, or more, but not all, of these points, the scheme is still intact.[30] Although it does not prevent an intruder from eventually succeeding, it does make it a more complex operation.

A third approach used by a number of software suppliers is to seed the TRM interrogation and subsequent compare instructions of a protection routine in a block of code, so that instructions comprising the routine are not readily identifiable. The intruder would be required to sift through each instruction in the block and decide whether or not it is part of the protection scheme. Furthermore, in some schemes, synchronous communication* between the CPU and the TRM allows sub-division of the interrogation instruction to the TRM. For example, the length of a serial number used by one microcomputer software supplier[31] is 16 bits. After the TRM has first been energised, or primed, each reading of its serial number requires 16 separate interrogation instructions. The first such instruction from the CPU to the TRM results in the transmission of the first bit of the serial number down one of the input/output port data lines. This is stored in memory. Later on, a second interrogation is sent, and the TRM replies with the second bit of the serial number. This also is stored. This process continues until all 16 bits have been transmitted to the

---

* Synchronous communication as interpreted here, means the transmission of <u>part</u> of a serial number from the TRM to the CPU, under program control without subsequent acknowledgement signals. This is distinct from asynchronous communication, in which, in this instance, upon interrogation, the whole serial number would be transmitted from the TRM to the CPU, followed by an acknowledgement signal.

CPU. In the intervening periods between interrogations, other data processing has occurred under program control. Therefore, the interrogating stage of the software protection routine comprises a number of instructions, all of which have to be identified by the intruder. Each is disguised by the other instructions of the block. At some later time, the distributed serial number is combined and compared with one residing in the program. If they agree, then processing proceeds. If not, then processing stops. However, should processing cease immediately after the comparing stage has discovered no serial number or the wrong serial number, then an intruder would know where this instruction resides in the program. Therefore, some companies incorporate a delayed failure routine to disguise when the COMPARE instruction is executed.[32]

A number of software suppliers use each of these measures, or variations of them, to minimise the probability of a successful software-based attacks.

Location of Hardware-based TRMs

Many of the more sophisticated TRMs each contain, at least, one microprocessor. In order to drive the microprocessor and associated circuits within the TRM, an external power source is required. Interrogation of the device now requires a preliminary energising, or priming, stage. This additional stage increases the protection of the program, for an intruder now has an extra layer of instructions to discover and by-pass.[33]

However, it also restricts the choice of input/output port, for a priming stage necessarily requires an interface including ground and power lines, as well as data lines.

On any particular type of microcomputer there are only a limited number of interfaces, some of which are reserved for particular peripheral devices.* However, as yet, no microcomputer has an interface specifically reserved for a protection peripheral.

The choice of interface, or port, depends primarily on the hardware architecture of the microcomputer, and this varies from one manufacturer to another. It also depends on the expected use of the interfaces by a typical end user of the software supplier. In general, a poor choice of interface can severely restrict the flexibility of the microcomputer system for the end user, for rarely can a port be shared between two peripheral devices at the same time.# For example, if an end user cannot interface a printer to his microcomputer because the software supplier requires the TRM to be located at that particular port, the user is clearly restricted in what he can do with his programs. Yet, the printer interface may have certain advantages in the supplier's software protection scheme compared with

---

* For example, the Apple II microcomputer has 8 parallel input/output interfaces, or slots. Slot 1 is reserved for a printer peripheral, slots 4 and 5 for input and output of data respectively.[34] In general, there are few spare slots.

\# A number of software suppliers have devised TRM devices that permit a peripheral device to be attached to it, so that the valuable interface is not taken up by the protection device.[35]

other interfaces he could use. Thus, in some cases, the choice of interface involves a trade-off between protection for the supplier and flexibility for the end user.

One company, marketing an accounting suite of programs designed to run on the Apple II microcomputer, decided to use the 'Games' input/output slot of this hardware to interface its TRM.[36] This interface provided a serial connection* between the TRM and the host microcomputer. One of the reasons for choosing it was that it was not considered useful to their particular end users. They reasoned that using it would not unduly affect the flexibility of their end user's microcomputer systems. However, the scheme is vulnerable to a traffic analysis attack on the relevant input/output lines. This is considered in the following subsection.

For Sirius I microcomputers, each has two parallel user ports, an external one reserved to interface the end user's printer with the host, and an internal one originally designed to interface other Sirius I microcomputers, to form a computer network. One software supplier, marketing a database management suite of programs, considered that their end users were not likely to require the internal interface, and therefore designed its TRM to be connected to it.[37] A further advantage is that the internal port is located inside the microcomputer chassis behind

---

* A serial connection, or serial port, is an interface in which data is transferred using one line only. In contrast, a parallel user port is an interface in which data transfer is achieved using two or more lines.

a steel plate. It was considered less likely that end users would tamper with the device. Each Sirius I microcomputer also has a single RS232C (or V24)* serial interface. This port has a 12V power source, and is used by one company to interface its protection device.[39]

For Commodore microcomputers, and particularly the Commodore PET, its cassette ports were chosen by a number of suppliers to interface their Dongles. These ports were originally designed to interface programs stored on magnetic tape. But, use of the ports for this purpose is no longer required, for many programs are instead supplied on floppy disks. The advantage of the cassette port is that it has a 5V power line, ground line, and 4 data lines providing a serial connection between the Dongle and the host microcomputer. The primary disadvantage of using a cassette port is that it is vulnerable to a traffic analysis attack.

Measures designed to foil Traffic Analysis Intruder Techniques

A typical traffic analysis attack comprises attaching a probe# to the lines connecting the peripheral to the host and

---

* The RS232C (or V24) port is a standard communications interface, comprising 25 lines. Each line has a specific function, the functions including ground, power, transmit data, receive data, handshaking and timing control lines. Most of these functions are not required in microcomputer data processing and typically only 3 lines are connected.[38] This implies that data transfer between the CPU and the TRM is serial, for 2 must be power and ground lines respectively.

\# The 'probe' can be a physical device, but not necessarily so. It could instead be a program which interrogates the protection device and monitors and analyses its response, or a 'cuckoo' module.

monitor communications along them. From this data, an intruder could identify the sequence of signals used to interrogate the protection device, and its response. The successful intruder could develop a 'pseudo-device' which would be able to correctly respond to particular interrogation signals from the host, fooling the program into believing that a proper device is attached. Thus, instead of using a vulnerable serial interface, a number of software suppliers have devised TRMs which interface with a parallel user port of the Commodore 4000/8000 series microcomputers.[40]

The primary advantage of a parallel interface is the speed of data communication between elements of the microcomputer system. In a given time, depending on the number of lines, much more data can be transferred. In the context of software protection, this implies greater application of encryption techniques to disguise communication, without serious reductions in the speed of data transfer inherent in a serial interface. However, if a parallel user port is used to interface a TRM, it can also severely decrease the flexibility of the microcomputer system for the end user. Thus, one of these suppliers[41] has devised a TRM which allows a peripheral to be attached 'piggy-back', to it. As a result, it does not take up one of the valuable parallel input/output ports and the end user can use it to interface one of his other peripherals, through the device.

This TRM contains two microprocessors, associated registers and memory. It is, in effect a microcomputer in its own right. When a signal is received by the TRM from the host microcomputer,

the program stored in the TRM analyses it and decides whether or not it is an interrogation for it, or a signal for the other peripheral. If it is destined for the 'piggy-back' peripheral, the signal is re-transmitted to it. Thus, from the end user's point of view, the TRM is transparent, for communication between the host microcomputer and the peripheral is virtually unimpeded. If, instead, the signal is an interrogation for the TRM, it replies with a suitably encoded acknowledgement signal.

In a typical TRM interrogation sequence, the protection routine of the program stored in unprotected memory decodes the instructions immediately prior to transmission, re-encodes it and transmits the re-encoded signal to the TRM. It decodes the signal, processes the interrogation, encodes a reply and transmits it to the host. The protection routine decodes the response and reacts accordingly. One set of keys for encoding and decoding the signals need not necessarily be used in all communications between the protection routine and the TRM. Rather, different keys can be used at selected points in data processing. Thus, should an intruder attempt a traffic analysis attack, predicting TRM responses from the collection of signals obtained, is difficult, for different responses from the TRM are obtained from similar interrogation signals from the host. The probability of fabricating a similar TRM from the data obtained in a traffic analysis attack is decreased.

For many software suppliers this level of protection is not necessary, for the services of a cryptanalyst would be required to break it. Rather, currently at least, a less rigorous software protection scheme is sufficient.

Necessity for Individually Distinguishable TRMs?

Most software packages comprise a collection of modules. Each module is a program designed for a particular application. For example, accounting packages normally include sales ledger, purchase ledger and nominal ledger modules. It is usual for all to be contained on the one disk. A typical end user requires some of these modules but not others. Therefore, the supplier requires a software protection scheme permitting use of the modules the end user had licensed, but not use of the remaining modules on the disk. The scheme must, in effect, 'lock out' these other modules. A number of software suppliers have developed similar schemes independently,[42] and each operates as follows: a request for access to a module is first processed by the protection routine residing in the package. It compares the name of the requested module with the one, or ones, residing in the TRM. If a certain response is received from the TRM, indicating that the end user is permitted use of the module, the protection routine allows access to it. On the other hand, if the response indicates that the end user is not allowed to use the module, the protection routine prohibits access. If follows that individually distinguishable TRMs are no longer necessary, for end users requiring the same combination of modules are supplied

identical devices. This is a significant advantage, for the software supplier can manufacture these TRMs in batches. Should an end user subsequently require an additional module, or modules, a different TRM is needed to permit data processing using the modified combination of modules.

Software Protection Products

Each of the techniques described above is used to protect particular suppliers' own programs. Similar techniques can also be used to protect other suppliers' programs from unauthorised use. A desirable feature of any software protection product is if it combines its primary protection function with another important function. One example is the DTL compiler.[43] Apart from the normal compiler processing function, it also inserts security routines at selected points within the compiled programs. The security routines tie the compiled program to a TRM containing a serial number. This particular compiler runs on Commodore hardware having two external cassette ports although similar schemes are now under development for other manufacturers' hardware.

In operation, it uses both ports. A dongle, to protect the compiler from unauthorised use, is plugged into one of the ports. The TRM to which the resulting compiled programs will be tied, is attached to the other port.

The company supplying the compiler, also manufactures the TRMs. Each software firm buying, or licensing, a copy of the

compiler has a serial number assigned to it, this being incorporated into its TRMs. The software firm supplies each of its customers with a copy of the compiled program together with a TRM required to use it in data processing. The software firm controls the distribution of its TRMs to its customers and therefore the dissemination of its programs in the market.

COPY-PROTECT MEASURES

A disadvantage of many of the use-protect measures described above is their visibility to end users. The end user must attach the TRM to the appropriate interface in order to use his copy of the protected program. Copy-protect measures, on the other hand, are unobtrusive. The protection scheme is contained within the software. Many end users are probably unaware that they are using copy-protected programs. The protection provided by copy-protect schemes is different in nature from that provided by use-protect schemes. All the measures described below are incorporated into copy-protect schemes designed either to prevent the end user from being able to copy the program at all, or severely restrict the number of useful copies he can make. In some schemes, the prohibition on copying is confined to one particularly valuable program, whereas in others, end users are not permitted to copy any of the programs comprising the suite.

In common with most use-protect schemes, many copy-protect schemes are devised in-house. Details of these schemes are, for

obvious reasons, regarded as company confidential. As a result, there is no universally adopted copy-protection technique, rather each scheme has its own particular features. However, there are also a number of companies marketing copy-protect software products,[44] i.e. products designed to insert a protection routine(s) into software firms' programs so that the latter's end users are prevented from copying them. Thus some software firms use the same scheme because they have bought-in protection from the same supplier.

All copy-protect schemes allow the end user to run the protected program on the relevant hardware and operating systems software. This implies that each scheme must permit the operating systems software access to the program, or selected ones in a suite of programs. However, operating systems programs are not designed to protect programs implemented by them. Furthermore, all operating systems software contain utilities that are essential tools for any would-be intruder. For example, all contain programs designed to copy other programs.* Once an unprotected program has been translated from the disk, or tape as the case may be, to the computer memory,# the operating system 'monitor' or 'executive' programs can be used to examine, and/or modify, it. Monitor software typically include programs having the following functions:

---

* For example, CP/M 'PIP' command, Apple DOS 'SAVE' command, Commodore DOS 'BACKUP', 'DUPLICATE' and 'COPY' commands.[45]

#• This process is called 'down-loading'.

"(1) to examine and/or modify contents of memory locations;
(2) to examine and/or change the contents of any register;
(3) to store (dump) programs ...;
(4) to sequentially list the contents of memory locations between two given addresses."[46]

From the end user's point of view a monitor is an extremely useful utility should he wish to write his own programs. However, from the supplier's point of view, it creates problems, for it could be used to discover and disable the protection routines incorporated into his programs.

All copy-protect schemes comprise measures which are designed to prevent copying if, and only if, the program has a certain residence. Some measures counter particular threats when it resides on magnetic disk, whereas others apply when it has been down-loaded and resides in memory. Most of the former disk-based measures involve modifying the format of information contained on the disk, so that it (the information) is not in a form 'recognisable' by the relevant standard operating systems software or disk operating systems software (DOS) as the case may be. Therefore the program cannot be down-loaded, and run by the operating systems software. Use of the protected program requires a non-standard 'loader', a program designed to translate it from disk to memory. But, once residing in memory, the program is vulnerable for the reasons outlined above. Thus, many copy-protect schemes include a protection routine, or routines, designed to counter particular threats to the program when it resides in the Random Access Memory (RAM) of the microcomputer.

The copy-protect features described below are specific to certain hardware and operating systems software.

Disk-based Methods of Protection

The arrangement of information on all magnetic floppy disks and diskettes is similar, regardless of their size.[47] Each disk is divided into a number of 'tracks'. For an 8" disk 77 tracks are usual. Each track is, in turn, divided into 'sectors'. The number of sectors is variable, but typically 26 comprise a track. Each sector is, in turn, divided into a number of 'bytes', typically 256 and each byte comprises a set number of bits. Byte sizes are variable, with 8 or 16 bits being typical. Information contained in the disk is arranged in a format comprising 'header' and 'data' sections. Each sector contains a header section and a data section. The header contains information identifying the particular track, sector and other data required by the DOS to transfer the data section from the disk to the microcomputer's RAM. A header typically contains the following: synchronization bits, track number, sector number, a checksum bit string, further synchronization bits and, perhaps an additional identification bit string.[48] Each of these items of information is required by the DOS to ensure that the correct track and sector has been accessed, and that the data contained therein has not been previously corrupted. The data section

follows the header, and contains, in this case, the program requiring protection. Protection measures can be incorporated at the sector, track and file levels.

At the sector level, should any of the data in the header be changed, then, on reading the sector, the standard DOS would interpret it as a corrupted sector. The sector would not be down-loaded, rather an appropriate disk error message would be displayed. Thus, a simple method of protecting sectors of the disk is to change the data contained in the header, for example, the track number, or the checksum bit string.[49] A more sophisticated method, of one software supplier, is to scramble the header and data sections together.[50] Both methods effectively prevented unauthorised down-loading of the valuable program when the relevant DOS is used alone. In each case, a non-standard loader program is required both to unscramble or 'uncorrupt' the sections, and down-load each data section therein.

At the track level, one software supplier changed the location of part of each track from that expected by the DOS. In operation, it would read part of the track, but the rest would be considered as corrupted data. The remaining unread part of the track would be located elsewhere on the magnetic disk.[51] Again, a non-standard loader would be supplied to authorised end users so that they could use the protected program.[52]

Each program is contained in one, or more, sectors. The group of sectors containing a program is called a 'file'. One of the tracks on the disk is reserved as an Index. Each entry in

the index specifies the track and sector numbers of the first sector of a file on the disk. Normally, should an operator require a particular program, the DOS first accesses the index, and subsequently the first sector of the file containing the program he wants. This sector contains a 'pointer' indicating the address of the second sector of the file. The second sector, in turn, contains, a pointer to the third sector of the file, and so on, except the last one of the file. Thus, each program on the disk comprises a chain of sectors, and access to one leads on to others in the chain.

One method of protecting a file is to insert a pointer in the last sector of the chain, to the first sector.[53] Thus the chain becomes a ring. Unauthorised down-loading is prevented for the DOS knows neither where the program begins nor ends. A second method used, is to delete the address of the first sector of the valuable program from the index, and restrict access to it by using an access protection routine.[54]

Each of the above methods of protection is effective in preventing unauthorised down-loading if an intruder uses only the relevant DOS or operating systems software. There are, however, software products on the market which can be used to overcome these and other copy-protect measures, if applied in conjunction with the relevant DOS or operating systems software.

Wiping the address of the first sector of a valuable program from the disk index was effective until early 1981. At that time, software products designed to copy sectors, rather than

files, were introduced onto the market.[55] This implied that end users no longer needed to consult the index in order to copy the relevant sectors. Wiping the sector from the index made no difference, for access to it could be otained using the relevant sector-copier software product.

By Spring 1981 bit-copying software products were introduced.*[56] These made few assumptions of the format of information contained in a sector. Effectively, encoded sectors could now be copied. Many of the currently available software protection products are designed to defeat copying using such products.[57] For obvious reasons, interviewees were prepared neither to specify in detail the techniques used to overcome these products, nor the techniques under consideration.

The disk-based methods of protection described above are designed to prevent unauthorised down-loading of the 'protected' programs. None of these methods modifies the operating systems software. Thus, once they reside in the microcomputer's RAM, they become vulnerable, for the protection provided by their non-standard format on the disk no longer applies.

Techniques for Protecting Programs stored in Memory

Each of the techniques for protecting programs residing in memory require the use of a protection routine incorporated into

---

* For example, 'LOCKSMITH', 'BACK-IT-UP', 'COPY-WRITE' and 'COPY II-PLUS'.

the program. The protection routine effectively lies between it and the operating systems software. Depending on the threat to the program, it either modifies the systems software in some way, or monitors it and reacts when a certain 'prohibited' operation is attempted.

Normally, down-loading and running a program in a micro-computer require two separate instructions. Once the program has been down-loaded into the computer memory, control is returned to the operator who then types in the appropriate RUN command to execute the software. But, he need not necessarily always RUN the program. He could, for example, copy it ('dump') onto another disk using the appropriate systems software instruction. Thus, many protection routines combine the two operations into one, so that once the down-loading instruction has been invoked, the program is also run automatically.[58]

An intruder could attempt to overcome this 'autorun' feature by 'interrupting' data processing using the standard interrupt utilities of the operating systems software. In a typical interrupt sequence, the CPU first completes the current instruction, and then stores the contents from the CPU's various registers (e.g., Program Counter, Accumulators, Instruction Register, Memory Address Register and General Purpose Registers) in a certain order, in a reserved portion of memory called a 'stack'. This stores the current condition of the CPU, so that once the interrupt has been serviced, the CPU can continue processing from where it has left off.[59] Once the contents have been stored,

the CPU can be used to service the interrupt. Thus, an intruder could branch to the operating systems software COPY utility. Thus, one method of preventing interrupts is for the protection routine to disable the stack.[60]

All microcomputers contain various interrupt input lines, one of which is called the interrupt-request line.[61] Interrupts are serviced if, and only if, the interrupt flag in the CPU's Condition Code Register* is set to a particular condition (e.g., the particular bit has binary state 'I'). A second method of preventing interruptions in processing is for the protection routine to set, or reset, the interrupt flag so that the CPU will ignore such requests,[63] (e.g., to ensure that the particular bit always is in condition binary state '0').

In every copy-protect scheme, the end user's ability to copy the program using the standard copy utilities of the particular systems software must be controlled by the software supplier. Otherwise there would be no restriction on the number of useful copies an end user could make of it. In some copy-protect schemes, COPY, BACK-UP and DUPLICATE commands are ineffective from the combination of the non-standard disk format, the 'autorun' feature, and the measures taken to prevent interruptions in data processing described above.[64] Other copy-protect schemes incorporate features designed to temporarily disable the copy utilities.[65] These are restored once the program has been run,[66]

---

* The Condition Code Register is a collection of bits called 'flags' indicating various conditions within the CPU.[62]

and its memory image scrambled. Some schemes are more selective, and permit copying of some programs but not others contained on the disk.

In one such scheme, the headers of the sector containing the valuable programs are encoded. When the software is downloaded, the operating systems first reads the protection routine, which specifies the permitted operating system utilities which can be used to implement the valuable program. It also supplies a key (or keys) for decoding the relevant headers. In operation, the protection routine monitors the error messages of the operating systems software. In this particular version of the operating system these messages are sufficiently specialised so that, if a particular error message is detected by the protection routine, it indicates that it had attempted to copy a sector having an encoded header, and failed. It had failed because it did not have access to the relevant decoding key. The attempted operation was not included in the permitted utilities of the protection routine. An approprate error message would be displayed. This protection routine does not prevent the end user from copying programs having plaintext headers.[67]

Some copy-protect schemes temporarily modify the standard COPY utility. In one such scheme, end users are allowed to 'copy' the disk once, but in so doing they also change both versions. The copy is not an exact reproduction of the original, and the protection routine is designed so that subsequent attempts to reproduce either the original, or the copy, fail.[68] A

variation of this approach is to allow copying of the program, but the copy will not run because it is not an exact replica of the original. This scheme depends upon the protection routine being able to detect whether or not the down-loaded program is a copy of the original supplied. If it is an original then the routine allows processing to proceed. If not then it branches to a stop and catch fire routine, thereby preventing data processing.[69]

MEASURES DESIGNED TO DETER UNAUTHORISED COPYING AND USE

Both categories of measures considered above are designed to prevent a user from either copying or using the program without prior authorisation. In contrast, the measures described below rely upon his honesty. They do not prevent him from copying or using the program, rather they are designed to deter him from so doing.[70] Thus, unlike the previous categories, the effectiveness of these measures depends upon their visibility. In theory, the greater the visibility, the less likely the end user, or dealer, would abuse the supplier's legal rights in his program.

The measures can take the form of a claim to the copyright in the software. A copyright notice including the name of the copyright owner, the year of publication and the word 'Copyright', 'Copr.' or '(c)', are written into many programs.[71] When an end user down-loads and/or uses the program in data processing, this notice is displayed so that he is in no doubt as to the supplier's

claim. He may be reminded of it at selected points in processing. For example, in some suites of programs, a copyright notice is displayed upon each return to the program MENU.[72]

Some suppliers tie copies of their programs to their authorised end users. One method used is to insert the authorised end user's name at selected points in his copy.[73] The name is displayed when it is run. In one scheme, all printouts are headed by the name and address of the authorised end user.[74] In each case, the measures are designed so that there can be no doubt as to the rightful owner of the copy.

Other measures include the insertion of serial numbers at selected points in each copy, and on its sleeve.[75]

The effectiveness of each of these measures also depends upon the ease, or difficulty, with which such notices can be removed. Clearly, a notice does not provide much protection if an intruder can easily patch over it, or prevent its display on the end user's screen. Thus, many software suppliers insert notices at various points throughout the program, so that should an intruder discover and remove one, or more, but not all, of them, the protection is still intact.

Other suppliers incorporate measures designed to detect modifications to the program. One technique used is to tie each instruction of the program to a particular line.[76] Should a copyright notice be removed this would be detected by the protection routine since the location of a particular instruction in the modified program would be different from its location in

the original. A second method used is to construct a serial number from selected bits of the program. The constructed number is compared with a corresponding one residing in it. If an intruder had modified the program, then he may have changed one or more of the bits used to construct the serial number. If so, then this would be detected when it is compared,[77] and a suitable error message displayed.

SUMMARY

Each of the measures described above is included in one, or more, microcomputer software protection schemes. These schemes are designed to aid software suppliers' control of the dissemination and use of their programs in the market. There is no single universally-applied software protection scheme, rather different ones are used by different suppliers. But, on the basis of 31 interviews and the survey of 156 software firms, reported previously, current software protection schemes can be classififed as follows:

(a) schemes designed to prevent unauthorised use of programs;

(b) schemes designed to prevent unauthorised copying of programs; and

(c) schemes designed to deter unauthorised copying and use of programs.

Every measure, in a scheme, is designed to counter a particular threat to the supplier's control of the dissemination and use of his program.

The categories are not mutually exclusive for certain measures designed to control access to programs, when they reside in RAM, have been applied in both use-protect and copy-protect schemes. categories (a) and (b) above. Furthermore, measures designed to detect unauthorised modification of programs are included in each category.[78]

Many use-protect schemes contain measures designed to tie a copy of the program to the end user's microcomputer hardware or operating systems software. This usually involves attaching, or inserting, a distinguishing feature into either the operating systems software, or the hardware. Data processing using the product is permitted, if, and only if, that feature is present throughout. In a number of similar schemes, a hardware device containing a serial number must be attached to the relevant interface. The serial number is compared with a corresponding one residing in the program at selected points in data processing. If the wrong device has been interfaced, this would be detected by the routine in the program comparing the two numbers. In each case, the protection scheme ties the program to one, or a small number of, similar microcomputers. There are many variations of, and refinements to, this general scheme, and particularly the application of measures designed to deter/prevent particular intruder techniques. These include measures designed to disguise the various protection routines stored in RAM; the use of scrambling and other data protection techniques to secure communications between the device and the CPU; encapsulation of

the device in epoxy resin to prevent tampering of it; and restrictions on access to the program when it resides in the RAM of the microcomputer, or disk controller, as the case may be.

Copy-protect schemes include measures designed to prevent unauthorised down-loading of the program, and measures designed to restrict access to it (or parts therein for a suite of programs) when it resides in RAM. Each of the former disk-based measures involves changing the format of information contained in the magnetic disk, so that it (or particular tracks and/or sectors therein) is not 'recognisable' by the relevant DOS or operating systems software. Disk-based protection measures rely upon the DOS or operating systems software interpreting the protected sector as being corrupted. These measures include changing data contained in the header section, scrambling the header and data sections, and deleting the address of the first sector of the valuable program(s) from the disk index. In each case, authorised down-loading requires a non-standard loader program.

Measures designed to restrict access to the program, include combining the down-loading and execution commands into one, and temporarily disabling, or modifying, the copy utilities of the operating systems software.

Schemes designed to deter unauthorised copying and use rely upon their visibility to end users and intermediate dealers. The measures are psychological. They are not designed to prevent these activities, rather to deter end users or dealers.

These measures comprise copyright notices incorporated into programs, and/or notices identifying the authorised end user. The effectiveness of these measures depends upon both the honesty of the end user or dealer, and the ease, or difficulty, with which such notices can be removed. Thus, many schemes contain measures designed to make removal difficult. For example, many software firms seed each of their programs with notices so that removal of some, but not all, leaves the protection intact, for it is still displayed when the program is used in data processing.

CHAPTER 6

CONCLUSIONS AND FUTURE DEVELOPMENTS

Legal and Technological Protection for Programs

The protection of computer programs was, until the late 1970's, a matter of little practical importance to many software suppliers. Prior to the boom in sales of microcomputer hardware, applications programs were mainly written for one, or a small number, of end users, in each case the software being tailor-made to their particular requirements.[1] It was of limited, if any, use to others, apart from the computer user who commissioned it. There was no need to protect it, for there was little market for bootlegged copies. Systems programs, although they had a wider market than applications programs, were tied to particular computer hardware. Since the hardware could not easily be copied, it, in turn, protected this software.

The widespread availability of cheap microcomputer hardware in the late 1970's transformed the computing services industry and its hardware and software markets. The tumbling prices of computer hardware implied that computing was no longer restricted to those end users who could buy, or lease, expensive mini or mainframe computers. Small businesses and even individuals could afford to buy their own microcomputer systems. Thus, the number of computer users rapidly increased. Furthermore computer users were no longer data processing professionals.

Rather they were increasingly people who had little previous experience of computing. Microcomputer systems were designed so that anybody, with the minimum of prior instruction, could use them. End users may not know how the system worked, but they nevertheless had the ability both to use the program for their own data processing, and copy it, ostensibly for back-up purposes. In addition, the markets for software changed. No longer was software commissioned on a one-off basis, rather there now existed large markets for standard applications and systems programs.

However, the investment of time, money and expertise sunk into the development of each microcomputer program, meant that few software suppliers could afford to market their programs without also protecting them. Many microcomputer software suppliers feared that, without effective protection, their programs would be widely copied and distributed without their obtaining the revenue to which they were entitled.[2]

The traditional method of protecting programs is the non-exclusive licence or contract. It is common practice for both the end user, and the intermediate software dealer, to enter into a contractual arrangement with the software supplier, based on the latter's intellectual property rights in the software. Many executives of microcomputer software firms consider that this method of protection, is neither appropriate, nor effective, for their companies' programs.

If a company is marketing identical copies of a program to a large number of end users through a network of dealers, in the

event of unauthorised copying each end user and dealer is a possible source. It is simply not feasible for the software supplier to trace the source of the unauthorised copies from such a large number of suspects. The supplier knows that some of his end users, and perhaps dealers, are in breach of his licence but does not know who they are.

Also the number of unauthorised users of any program is unknown. Furthermore, it is unknowable, for reproduction of a program, authorised or unauthorised, from one disk to another sometimes requires merely use of the appropriate 'COPY' program in the operating systems software. Thus end users and dealers copy in private, with the software supplier being, in most cases, unaware that this is going on until well after the event. The legal counsel of one US-based microcomputer software house, marketing a popular wordprocessing suite of programs, estimates that at least four out of every five of his company's end users are not authorised to use it. He believes that his company is losing at least 80% of the software revenue to which it is entitled.[3] Most firms in the UK computing services industry are not losing revenue on this scale. Indeed, the survey results indicated that for most business, commercial and industrial software suppliers in the UK, instances of unauthorised copying and software piracy are not common. However, 20% of microcomputer software firms in the sample have incurred 'serious' losses of revenue from one, other or both of these activities.

Many of those found out are discovered by chance. If the

supplier then decides to prosecute these infringers, he is then faced with the problems of enforcing intellectual property rights which have yet to be established for the technology, let alone the programs copied.

Computer programs sit uneasily between the traditional types of work protected by copyright, and inventions capable of patent protection. Human-readable embodiments of programs (e.g. written source code listings) are probably copyrightable for they are similar to literary, and artistic, works which have long been regarded as suitable subject matter for copyright protection. However, without statute and/or case law it is uncertain whether machine-readable embodiments of computer programs, e.g., programs expressed on magnetic disks, and programs expressed in ROMs, are also copyrightable under the Copyright Act 1956. Furthermore, it is uncertain just how far the copyright owner's rights extend. In particular it is uncertain whether both machine-readable embodiments of programs, and the translations and transformations a program undergoes in running it in a computer, are included in the statute's restricted acts.

In the US, it is becoming established that human-readable and machine-readable embodiments of a computer program are capable of copyright protection.[4] The copyright owner's exclusive rights extend to all 'fixed' embodiments of the program, but not to the algorithm underlying it. It is uncertain whether the use of a program in a computer is included in his exclusive rights.

Patent Office examiners, both in the UK and US, have struggled to understand the nature of computer programs. Originally all patent applications for software-inventions were rejected on the grounds that they were either claims specifying merely mathematical formulae (considered unpatentable subject matter) or that the 'end product' in each application was merely 'data', and not a new, and useful, product, or an improvement to an existing product. However, it is now established that computer programs are excluded from patent protection only when claimed as such. Software-inventions claimed as computer-implemented industrial processes, and/or certain industrial products are capable of protection. However it is thought that most software-inventions are not sufficiently original for each to warrant a patent application. Patents can be a useful supplement to the supplier's copyright in his programs but it is rarely his main protection.[5]

The enforcement of intellectual property rights, and particularly copyright, is invariably expensive. Although this is not a problem unique to disputes involving computer programs, it has deterred many software firms from litigation. Furthermore, unless there is a good chance that a guilty defendant can pay the award of damages stipulated by the Court, it makes little commercial sense for a software supplier to enforce its rights.[6] As a result, under UK law, there are few precedents clarifying both copyrightable embodiments of programs, and the scope of the owner's exclusive rights. Furthermore, and perhaps more important, the lack of case law does nothing to educate the many

solicitors, barristers and judges who are not familiar with the technology. Many are not aware that the same program can have a multitude of different human-readable and machine-readable embodiments and conventionally undergo many transformations when it is run in a computer. Also, many software suppliers are not aware that litigation can proceed quickly. Interlocutory cases can be heard within a few days. Furthermore the court can, and does, act speedily to prevent the destruction of evidence prior to full trial, with the granting of Anton Piller orders. Rightly or wrongly, many executives of microcomputer software firms believe that the enforcement of copyright in court is too expensive, risky and slow to be worthwhile.

Many microcomputer software firms have supplemented their legal rights with technological protection measures written into their programs. Some of these measures are designed to aid the enforcement of copyright. Concealed copyright notices in the plaintiff's program and reproduced in the defendant's program would provide persuasive evidence of copying. However, other technological measures substantially replace the supplier's copyright, and are designed to <u>prevent</u> those activities that effective law would, in most instances, deter. While it is uncertain whether unauthorised machine-readable copies of a program constitute a copyright infringement of it, some suppliers take various measures designed to <u>prevent</u> end users and dealers from being able to copy it. While it is uncertain whether use of a program in a computer is a restricted act, some suppliers <u>prevent</u> unauthorised use by tying their programs to one, or a small number

of, authorised microcomputer systems. Thus, a perceived lack of effective legal protection, and particularly effective copyright law, has led to the development, and use, of these alternative technological forms of protection.[7]

However, the fear of widespread unauthorised copying and piracy, and the perceived lack of effective legal protection do not seem to have appreciably deterred investment.[8] Although they may have increased secrecy within the industry, the primary effect has been that some of this investment has been channeled into the development of technological methods of protecting software from unauthorised copying and use. Furthermore, it has given rise to a new type of software firm, the software protection supplier,* within an already heterogeneous industry. The products of these firms are designed to allow the software supplier to control the dissemination of useful copies of his program in the market. Some are designed to prevent end users from being able to make useful copies of the software. Other products are designed to prevent unauthorised use of the software. All of these firms owe their existence to the current uncertainties, and inadequacies, of copyright and patent law.

---

* Although some of the following companies also market other types of program, all supply software protection programs in their product ranges: SoftGuard Computer Systems, Computer Applied Technology Ltd., Mektronic Consultants, U-Microcomputers, TABS Ltd., Dataview Ltd., Little Genius Ltd., Wordcraft Systems, Sensible Software Inc. and Drive Technology Ltd.

Future Developments

In the survey (the main results of which were presented in Chapter 4 above) respondents were asked whether, in their view, innovations in computing will create, in their wake, problems in the protection of computer programs. One hundred and twenty-eight replies were received. Sixty-five respondents (50.7% of replies received) considered that protection problems will arise, and particularly from:

(a) developments in computer networks. Some respondents feared that the programs of one user, stored in the memory of his microcomputer could become accessible to other users of the network.

(b) The standardisation of operating systems software. It was feared that a potential black market for popular programs would be increased if the particular operating systems software for which they were designed, was compatible with hardware other than their authorised end users' microcomputers.

(c) The development, and availability, of 'decompilers'. These programs are an aid to analysing the structure of other programs, for they translate them from their compiled object code version to a source code form.. Some respondents considered that intruders could use a decompiler as a tool to either identify the protection routines of a program and subsequently by-pass them or recompile them in a different language to disguise the source of

the original.

The other 63 respondents (49.3%) considered that either no protection problems, or no additional protection problems to those already existing, will be created.

The technological protection of computer programs is likely to continue to be the main form of protection for many microcomputer software firms. These companies will develop their own protection measures to defeat, or deter, increasingly sophisticated bit-copying and other intruder techniques, as well as devise measures designed to alleviate some of the concerns expressed above. Furthermore, research into technological protection will be funded by the National Physical Laboratory, British Technology Group and also, it is hoped, from a consortium of UK software firms.

Other developments include a study, now being undertaken for member companies of the recently formed ADAPSO microcomputer software association, into legal and technological protection of computer programs.

Furthermore, as the computing services industry continues to grow, the need for effective technological protection will result in both an expanding number of software protection products, and companies specialising in developing them.

Developments can also be expected to clarify copyright law. In addition to the ever increasing body of overseas (and hopefully also UK) case law, the Copyright Committee of the British

Computer Society is pressing for an amendment of the Copyright Act 1956. The Committee's proposals are intended to clarify both the copyrightability of computer programs, and the scope of the copyright owner's rights. Since many of the uncertainties in interpretation of the 1956 Act stem from the inadequate definitions of Section 48(1), most of the Committee's proposals comprise the inclusion of new ones, that is, 'computer' and 'computer program', and additions to existing definitions. It proposes that the term 'computer' should be defined as:

> "'computer' means any device for storing and processing information;"[9]

and 'computer program' as:

> "'computer program' means any series of instructions which control or condition the operation of a computer."[10]

It proposes that the term 'writing' should be expanded as follows:

> "'writing' includes any form of notation whether by hand, or by printing, typewriting or any similar process [including notation expressed by mechanical, electrical, magnetic, chemical or other means]."*[11]

If this proposal is enacted it would establish that the copyrightability of a work extends to all machine-readable as well as human-readable embodiments. It also proposes that the definition of the term 'literary work' should be expanded to specify computer

---

* The squared brackets contain the proposed amendment.

programs.[12]

If all these amendments are enacted, it will establish that computer programs, in whatever language they are expressed, are capable of copyright protection as literary works.

The Committee's other proposals include the addition of a 'use right' in the restricted acts specified in Section 2(5), as follows:

> "The acts restricted by the copyright in a literary, dramatic or musical work are -
> .. ..
> [(ee) causing the work to be used to control or condition the operation of a computer]."*[13]

If this proposal is enacted, then it will establish that unauthorised down-loading and running a program in a computer would be an infringement of the copyright subsisting in the program.

Clearly the enactment of these proposals will substantially enhance the effectiveness of copyright. While it may deter some end users and dealers from illegally copying and using others' programs, it may not necessarily encourage software suppliers to enforce their rights in court. The other problems will remain. The cost of litigation, together with the problems of proving copyright infringement, mean that, for many microcomputer software suppliers, reliance will still be placed on their technological forms of protection. Copyright will be effective, if, and only if, microcomputer software suppliers

---

* The squared brackets contain the proposed amendment.

are prepared to enforce their rights in court. Some insurance companies now offer policies to software firms, under which, in the event of infringement of the firm's intellectual property rights, the insurance company pays for the costs of litigation up to an agreed limit. It is hoped that this will encourage software firms to enforce their rights in court. In addition, companies in the industry should collaborate and collectively prosecute persistent copyright infringers thereby spreading the costs of enforcement. It is hoped that such consortia will become established, although attempts to date have failed.

General Conclusions

The transformation of a secret invention to an innovation requires, amongst other things, informing an entrepreneur, product champion, publisher etcetera of details of it, or generally publishing it in an attempt to attract investment. Intellectual property rights encourage inventors to publish their work, for even though the information is no longer secret, they still retain certain rights in them, recognised in law. Intellectual property rights, in turn, encourage the product champion to invest in the invention, secure in the knowledge that rights in its development would be recognised in court. Patents and copyrights serve both functions for their respective types of work. In both cases, society benefits from the dissemination of protected knowledge.

However, the effectiveness of these arrangements breaks

down if the work is not protectable by intellectual property or allied rights (such as confidentiality), or if such rights as do exist cannot easily be enforced. For some inventors this may encourage greater secrecy, thereby depriving society temporarily or permanently of details of their creations. It may deter an entrepreneur investing in the work, for he would not be willing to risk losing his investment if his rights in it are not recognised in law. In both circumstances the invention will not become an innovation. However, in the case study, the lack of effective legal protection has had a quite different effect. It has encouraged the development of alternative, technological methods of protection, designed to prevent those activities that effective intellectual property rights would, in most instances, deter. Thus, while intellectual property law encourages the development of all protectable creations, the lack of effective law encourages a particular type of invention, designed to effectively replace the law for technologies in which intellectual property rights are either not clear or not easily enforceable.

APPENDIX

QUESTIONNAIRE

PROTECTION OF COMPUTER SOFTWARE

NAME ..... FIRM .....

POSITION .....

Please ring, or tick, whichever is appropriate.

1. Is unauthorised copying (i.e., private copying) of your software by END USERS common?

   a. YES
   b. NO
   c. DON'T KNOW

2. If YES, does it constitute a serious loss of business to your firm?

   a. YES
   b. NO
   c. DON'T KNOW

3: Does software piracy (i.e., the selling of unauthorised copies of your software products) by SOFTWARE DEALERS constitute a serious loss of business?

   a. YES
   b. NO
   c. DON'T KNOW

Legal Protection

4. Which of the following does your firm use to protect its software?

   a. Patents
   b. Copyrights
   c. Trade Marks
   d. Law of Confidence/Trade Secrets
   e. Licences/Contracts
   f. None of these

5. Is/are the chosen mode(s) of legal protection effective?

Always	Sometimes	Hardly Ever	Never

Protection from the Nature of the Product

6. Does the END USER's dependence on your firm for maintenance confer effective protection?

Always			Never

7. Does the investment of time and money spent in developing the software confer effective protection?

Always			Never

Contd./Qu.8

Technical Protection

8. Are technical measures taken to protect your computer programs?

a. YES
b. NO

YES → 9. Are these measures designed to:

i) prevent/deter copying of the programs?

a. YES
b. NO

If YES, please specify measures (e.g., uncopyable disks)

.....

ii) prevent/deter use of the programs?

a. YES
b. NO

If YES, please specify measures (e.g., passwords)

.....

iii) aid discovery/proof of the source of the copy?

a. YES
b. NO

If YES, please specify measures (e.g., software or hardware serialisation)

.....

NO → Proceed to Question 10

10. Were technical measures once taken but have since been abandoned?

a. YES
b. NO

11. Does your firm allocate resources for protecting its software?

a. YES
b. NO

If YES, please specify measures (e.g., employing an encryption expert)

.....

12. Is the protection of software a consideration in its design?

Always	Sometimes	Hardly Ever	Never

Contd./Qu.13

General

13. Is your firm:

a. An Applications Software House
b. A Systems Software House
c. A Software Publisher
d. A Software Distributor
e. A Computer Manufacturer
f. Other. Please specify

.....

14. How many professional and technical employees does your firm have?

a. 1 - 9
b. 10 - 99
c. 100 - 999
d. 1000+

15. In what year was your firm set up?

a. Pre 1965
b. 1965 - 1970
c. 1971 - 1975
d. 1976+

16. What type of software does your firm supply?

a. General Business Applications
(e.g., accounting, word processing)
b. General Financial Applications
(e.g., payroll, tax)
c. Engineering and Scientific Applications
d. Systems Software
(e.g., compilers)
e. Other. Please specify

.....

17. Which type, or types, of end user does your firm supply? Please rank their importance as markets for your firm's products on a scale of 1 - 4, where 1 = most important market; 2 = an important but not the most important, market; 3 = a minor market; and 4 = a market not supplied.

		Rank			
a.	Large companies	1	2	3	4
b.	Medium to small businesses	1	2	3	4
c.	Educational institutions	1	2	3	4
d.	Local government	1	2	3	4
e.	Central government	1	2	3	4
f.	Other end users. Please specify				
	.....	1	2	3	4

Contd./Qu.18

18. What price of hardware does your software run on?

a. up to £10,000
b. £10,000 - £30,000
c. £30,000 - £80,000
d. £80,000+

19. Will innovations in computing create protection problems?

a. YES
b. NO

Please comment .....

.....

20. Has a lack of legal protection influenced (or will influence) innovations in computing?

a. YES
b. NO

Please comment .....

.....

Any further comments:

- END -

Please return to: S. M. Elsom
Technology Policy Unit
University of Aston in Birmingham
Gosta Green
BIRMINGHAM
B4 7ET

REFERENCES TO INTRODUCTION

1. Christopher Freeman, The Economics of Industrial Innovation, Penguin, 1974, pp. 16-17; Edwin Mansfield, The Economics of Technological Change, Norton, 1968, pp. 8-9.

2. Freeman, ibid. (note 1).

3. Ibid., p. 22 (note 1).

4. Jacob Schmookler, Invention and Economic Growth, Harvard University Press, 1966, pp. 24-25.

5. E.W. Ploman and L. Clark Hamilton, Copyright: Intellectual Property in the Information Age, RKP, 1980, p. 22; "Copyright and Designs Law: Report of the Committee to consider the Law on Copyright and Designs", Chairman: The Honourable Mr. Justice Whitford, Cmnd. 6732, March 1977, (Reprinted 1980), paragraph 13, p. 3. Termed hereafter "Whitford Report".

6. "The British Patent System: Report on the Committee to examine the Patent System and Patent Law", Chairman: M.A.L. Banks Esq., Cmnd. 4407, July 1970, (Reprinted 1976), paragraph 1, p. 1. Termed hereafter "Banks Report"; Ploman and Hamilton, ibid., p. 24 (note 5).

7. Ploman and Hamilton, ibid., pp. 22-23 (note 5).

8. C.T. Taylor and Z.A. Silberston, The Economic Impact of the Patent System, Cambridge University Press, 1973, p. 25; Bryan Niblett, Legal Protection of Computer Programs, Oyez Publishing Ltd., 1980, p. 17.

9. E. Braun and S. Macdonald, Revolution in Miniature: The History and Impact of Semiconductor Electronics, Cambridge University Press, First Edition, 1978, p. 129; Freeman, op. cit., p. 185 (note 1).

10. Banks Report, op. cit., ch. 2, para. 41 (note 6).

11. Hugh Laddie, Peter Prescott and Mary Vitoria, The Modern Law of Copyright, Butterworths, London, 1980, p. 1, para. 1.1; Ploman and Hamilton, op. cit., p. 24 (note 5).

12. Ploman and Hamilton, ibid., p. 24 (note 5).

13. Whitford Report, op. cit., p. 5, para. 17 (note 5).

14. See Freeman, op. cit. (note 1); Mansfield, op. cit. (note 1); J. Jewkes, D. Sawers and R. Stillerman, The Sources of Invention, Macmillan, 1958, p. 135; and C. Freeman, "The Determinants of Innovation" in Futures, June 1979, pp. 209-10.

15. Schmookler, ibid., p. 56 (note 4); "Science and Technology Indicators Conference", Organisation for Economic Co-operation and Development (OECD), Paris, 15-19 September 1980.

16. Schmookler, op. cit. (note 4).

17. Schmookler, ibid., pp. 24-25, 55 (note 4); E.W. Kitch, "The Use of Patent Statistics in Science Indicators", Second Workshop on the Measurement of R&D Output, OECD Paper 1.4, 5-6 December 1979, pp. 57-61.

18. Richard I. Miller, Legal Aspects of Technology Utilization, Lexington Books, © 1974 D.C. Heath & Company.

19. Miller, ibid., p. 4 (note 18).

20. Miller, ibid., p. 55 (note 18).

21. Miller, ibid., p. 56 (note 18).

22. Miller, ibid., p. 56 (note 18).

23. Miller, ibid., p. 10 (note 18).

24. Miller, ibid., p. 4 (note 18).

25. Roy N. Freed, "Software Protection: Introductory Observations on the study sponsored by the National Commission on New Technological Uses of Copyrighted Works" in Jurimetrics Journal, vol. 18, no. 4, Summer 1978, p. 352.

26. R.I. Miller, "The CONTU Software Protection Survey" in Jurimetrics Journal, vol. 18, no. 4, Summer 1978, p. 367.

REFERENCES TO CHAPTER 1

1. Margaret L.B. Anderson, Software Protection: a survey of the industry, Final Report, University of Kent, August 1976, p. 27. Termed hereafter "Anderson Survey".

2. Louis Frenzel, Getting Acquainted with Microcomputers (Howard W. Sams & Co. Inc., Indianapolis, Indiana 46268, First Edition 1978).

3. Ibid., pp. 113-114.

4. Lawrence Perry and N.A. Killgren, "Computer Programs - Art or Science" in The Journal of the Chartered Institute of Patent Agents, December 1980, pp. 97-122.

5. Ibid., p. 100.

6. World Intellectual Property Organisation (WIPO) Model Provision on the Protection of Computer Software, Geneva 1978. Termed hereafter "WIPO Model Provisions".

7. Ibid., Section 1(i), p. 9.

8. To amend Title 17 of the United States Code to improve the Protection afforded to Computer Software, and for other Purposes, HR 6983, 97th Congress 2D Session (Kastenmeier Bill), 12th August 1982.

9. Hugh Brett and Lawrence Perry (eds.). The Legal Protection of Computer Software (ESC Publishing Ltd., Oxford 1981) p. 2; Bryan Niblett, Legal Protection of Computer Programs (Oyez Publishing Ltd., London 1980) pp. 9-10.

10. Niblett, ibid.

11. Frenzel, op. cit., p. 114 (note 2).

12. Ibid.

13. Interview with Simon Bennett, a freelance microcomputer programmer, on 12th March 1981.

14. Niblett, op. cit. (note 9).

15. Peter Calingaert, Operating System Elements: a user perspective (Prentice Hall In., 1982) p. 2. Calingaert follows Habermann's categorisation of systems programs. c.f. A.N. Habermann, Introduction to Operating System Design (Science Research Associates, Palo Alto, California, 1976).

16. Calingaert, ibid.

17. Ibid.

18. Frenzel, op. cit., p. 127 (note 2).

19. S.T. Kent, Protecting Externally Supplied Software in Small Computers (MIT Thesis, Sept. 1980) ch. 5.

20. Calingaert, op. cit. (note 15).

21. Niblett, op. cit., pp. 11-12 (note 9); Frenzel, op. cit., p. 115 (note 2).

22. Interview with Duncan Davidson, Chairman of the American Bar Association Subcommittee on software protection, 24th November 1982.

23. Ibid.

24. Ibid.

25. Ibid.

26. Niblett, op. cit.

27. Frenzel, op. cit., p. 124 (note 2).

28. Interview with Sandy Livingstone, Technical Manager, Taylor-Wilson Systems, 17th September 1982.

29. Frenzel, op. cit., pp. 121-123 (note 2).

30. Ibid., pp. 123-124 (note 2).

31. Ibid.

32. Ibid.

33. Interview with Duncan Davidson, op. cit. (note 22); Perry and Killgren, op. cit., p. 100 (note 4).

34. F.F. Mazda, Integrated Circuits: technology and applications (Cambridge University Press, 1978) pp. 115-117.

35. Frenzel, op. cit., p. 130 (note 2).

36. Niblett, op. cit., p. 13 (note 9).

37. Frenzel, op. cit., pp. 130-132 (note 2).

38. Ibid, p. 130 (note 2); see also Niblett, op. cit., (note 9).

39. Frenzel, ibid., pp. 134-135 (note 2); Niblett, ibid. (note 9).

40. Frenzel, ibid. (note 2); Niblett, ibid., p. 14 (note 9).

41. Frenzel, ibid., (note 2); Niblett, ibid., pp. 14-15 (note 9).

42. Frenzel, ibid., (note 2); Niblett, ibid., p. 15 (note 9).

43. Niblett, ibid., p. 16 (note 9).

44. Appellant's Opening Brief, Apple Computer Inc. v. Franklin Computer Corporation (US Court of Appeals for the 3rd Circuit, (82-1532, 82-1582) 23rd October 1982) pp. 6-7.

45. Niblett, op. cit., p. 13 (note 9).

46. WIPO Model Provisions op. cit., section 1(iv), p. 9 (note 6).

47. Ibid., section 1(ii), p. 9 (note 6).

48. Ibid., section 1(iii), p. 9 (note 6).

49. Frenzel, op. cit., p. 114 (note 2).

50. Julian Allason, "Maggie goes micro", Observer Business Supplement, 27th March 1983, pp. 17-20.

51. Ibid.

52. Interview with John Cartwright, Manager Group Patent Services, ICL, 14th May 1981.

53. Ibid.

54. Ibid.

55. Ibid.

56. Ibid.

57. Interview with Bryan Niblett, 27th July 1981.

58. Interview with John Cartwright, op. cit. (note 52).

59. Richard I. Miller, Legal Aspects of Technology Utilisation (D.C. Heath & Co., 1974) p. 56.

60. Richard I. Miller, "The CONTU Software Protection Survey" in Jurimetrics Journal, Summer 1978, vol. 18, no. 4, p. 358.

61. ADAPSO Annual Report, 1981, p. 1.

62. Interview with Ron Palenski, Assistant General Counsel, ADAPSO, 29th November 1982.

63. Anderson Survey, op. cit., p. 3 (note 1); CSA Directory of Members 1982.

64. CRA Computer Buyer's Guide 1982-83.

65. Alan Cane "Concern over the restraints of Government" in Computing Services and Software IV, Financial Times, 5th November 1982.

66. Ibid.

67. Frenzel, op. cit., pp. 12-14 (note 2).

68. P.J. Miller, "The Microprocessor and the Microcomputer", Microprocessor Systems Design, 3-day introduction to microprocessor engineering, 9-11 March 1981, Microprocessor Unit, University of Aston, First day, p. 2.

69. D.J. Holding, "Introduction to the Microprocessor", Microprocessor Systems Design, 3-day introduction to microprocessor engineering, 9-11 March 1981, Microprocessor Unit, University of Aston, First day, p. 4.

70. Ernest Braun and Stuart Macdonald, Revolution in Miniature: the history and impact of semiconductor electronics (Cambridge University Press, 1978; second edition, 1982).

71. P.J. Miller paper, op. cit. (note 68).

72. Phil Manchester, "Chance comes up trumps with CP/M" in Computing, 11th November 1982, p. 36.

73. Interview with Gervaise Davis III, Legal Counsel, Digital Research Inc., 22nd November 1982.

74. Ibid.

75. Ibid.

76. Ibid.

77. CSA Guidelines for Software (Program) Product Licences, 1978.

78. Submission by the Computing Services Association in Response to the Consultative Document in the Reform of the Law Relating to Copyright, Designs and Performers' Protection (Green Paper, Cmnd. 8302) 3rd February 1982, pp. 4-5.

79. Interview with Barney Gibbons, Managing Director and Chairman, CAP-CPP, 8th October 1981.

80. Ibid.; interview with Sid Newman, Dataview Limited, 19th February 1982.

81. Interview with Dave Sheppard, Marketing Manager, Datacall Limited, 2nd September 1982.

82. Interview with E. Ric Giardina, Legal Counsel, MicroPro International Inc., 17th November 1982.

83. Interview with Ian Monro, Managing Director, Computerised Business Systems Ltd. (CBSL) 6th August 1982.

84. Interview with E. Ric Giardina, op. cit., (note 82).

85. Ibid. (note 82); Sean Hallahan, "Using the law to beat the pirates" in Computing, 8th April 1982, p. 10.

86. Interview with E. Ric Giardina, ibid. (note 82).

87. Ibid. (note 82).

88. Sean Hallahan, "A long search for effective deterrents" in Computing, 24th June 1982, p. 23.

89. Ibid.; Claire Gooding, "Pirates take most of micro software trade" in Computer Weekly, 12th March 1981, pp. 1 and 9; Jeffrey C. Brown, "Everybody does it - software pros sound off about piracy" in InfoWorld, 22nd March 1982, p. 39.

90. Interview with E. Ric Giardina, op. cit. (note 82).

91. Ibid.

92. Sean Hallahan, op. cit. (note 88).

93. "Price on pirates' heads", Computer Dealer, Jan 1983, p. 1.

94. Interview with Lynn Brock, Visicorp Inc., 19th November 1982.

95. Interview with E. Ric Giardina, op. cit. (note 82); interview with Lynn Brock, ibid. (note 94); interview with Sid Newman, Dataview Limited, 2nd March 1982.

96. Interview with E. Ric Giardina, op. cit. (note 82); interview with Lynn Brock, ibid. (note 94).

97. Richard Waller "Up the Sharp End: a user's view of software piracy", Personal Computer World, October 1981, p. 132.

98. Barney Gibbons, "Software Protection by Licensing", Workshop Panel No. 10, World Computing Services Industry Congress III, 22nd June 1982, Copenhagen, Denmark.

99. Interview with Sid Newman, op. cit. (note 95).

100. e.g., Paul Gillin, "Judge Awards $250,000 to Software Developers" in Computerworld, 27th September 1982, p. 106; Paul Walton, "Vector set to test piracy law in UK" in Computing, 7th October 1982, p. 7.

REFERENCES TO CHAPTER 2

1. "Taking the Software Pirates to task", Which Computer?, September 1981, pp. 109, 111-112.

2. See for example, Bryan Niblett, Legal Protection of Computer Programs (Oyez Publishing Ltd., London 1980), Coda p. 111; Joseph B. Taphorn, "New Developments in Copyright Protection of Computer Programs" in Communications and the Law, Vol. 1, No. 3, Summer 1979, p. 29 at p. 44; and Final Report of the National Commission on New Technological Uses of Copyrighted Works (CONTU), published 31st July 1978, Ch. III C (c) at p. 44. Termed hereafter "CONTU Report".

3. L.S. Nixon, "Protection of Intellectual Property Rights in the United States for Computer-aided Processes, Machines and Audio-visual Displays" in the Journal of the Chartered Institute of Patent Agents, Vol. 12, No. 8, May 1983, p. 349.

4. Copyright Act, 1956, 4 and 5 Eliz. 2, Ch. 74.

5. John Cartwright (Group Patents Manager, ICL), "How to identify the Infringer?", a presentation given, on 24th May 1983, at a meeting of the Technology of Software Protection Specialist Group of the British Computer Society, 13 Mansfield Street, London, W1M 0BP.

6. Copyright Act, 1956, op. cit., p. 2, (note 4).

7. Gates v. Swift and Others 1982 RPC 339; Sega Enterprises Ltd. v. Alcan Electronics and Others 1982 FSR p. 516; Systematica v. London Computer Centre 1983 FSR p. 313; Martin Hayman, "Software Protection in the United Kingdom", Byte, November 1981, p. 126.

8. Martin Howe, "Protection of Computer Software", Polytechnic of Central London, School of Law, Industrial Property Rights Course, 23-24 April 1982.

9. See, for example "Copyright", Copinger and Skone James, 12th Edition, para. 154, p. 60; The Report of the Committee to consider the Law on Copyright and Designs, (HMSO, Cmnd. 6732) p. 133, para. 520(i), Chairman: the Honourable Mr. Justice Whitford. Termed hereafter "Whitford Report".

10. Niblett, op. cit., Ch. 4, pp. 42-43 (note 2).

11. Whitford Report, op. cit., p. 4, para. 16 (note 9).

12. Ibid., p. 5, para. 17 (note 9).

13. Copyright Act, 1956, op. cit., Section 48(i), p. 64 (note 4).

14. Ibid., p. 65 (note 4).

15. H. Laddie, P. Prescott & M. Vitoria, The Modern Law of Copyright, (Butterworths, London, 1980) pp. 11-12, paras. 2.10-2.11.

16. Boosey v. Whight (1899) 1 Ch. 836, 842.

17. Laddie et al., op. cit. (note 15).

18. Copyright Act, 1956, op. cit., Section 49(4), p. 68 (note 4).

19. Reform of the Law relating to Copyright, Designs and Performers' Protection. A Consultative Document. Presented to Parliament, July 1981. (HMSO Cmnd. 8302). Termed hereafter "Green Paper".

20. Green Paper, ibid., Ch. 8., p. 33, para. 2 (note 19).

21. Green Paper, ibid., Ch. 8, p. 33, para. 3 (note 19).

22. Copyright Act, 1956, op. cit., Section 2(1), p. 2 (note 4).

23. Whitford Report, op. cit., p. 10, para. 33 (note 9).

24. H. Laddie et al, op. cit., pp. 23-24, paras. 2.36-2.37 (note 15).

25. Niblett, op. cit., Ch. 4, pp. 43-44 (note 2).

26. Copyright Act, 1956, op. cit., Section 3(1), p. 4 (note 4).

27. Laddie et al., op. cit., p. 95, para. 2.143 (note 15).

28. Copyright Act, 1956, op. cit., Section 2(5), p. 3 (note 4).

29. (1964) 1 WLR 273 at p. 291; Niblett, op. cit., Ch. 4 p. 14 (note 2); W.R. Cornish, Intellectual Property: Patents, Copyright, Trade Marks and Allied Rights, (Sweet and Maxwell, 1981) p. 306.

30. Copyright Act, 1956, op. cit., Section 48(1), p. 64 (note 4).

31. Ibid., Section 2(6)(iii), p. 3 (note 4).

32. Sega Enterprises Ltd. v. Richards and Another 1983 FSR Part 2, p. 73 (2nd July 1982) at p. 75.

33. Green Paper, op. cit., Ch. 8, pp. 33-34, para. 4 (note 19).

34. Whitford Report, op. cit., p. 133, para. 520(v) (note 9).

35. Green Paper, op. cit., Ch. 8, p. 34, para. 5 (note 19).

36. "Taking the Software Pirates to task", Which Computer? op. cit., p. 111 (note 1).

37. Copyright Act, 1956, op. cit., Section 49(1), p. 67 (note 4).

38. Reproduced in Cornish, op. cit., Ch. 11, pp. 347-348 (note 29).

39. Copyright Act, 1956, op. cit., Section 3(5), p. 5 (note 4).

40. Laddie et al., op. cit., p. 95, para. 2.142 (note 15).

41. Whitford Report, op. cit., pp. 185-186, para. 728 (note 9).

42. Laddie et al., op. cit., p. 48, para. 2.74 (note 15).

43. Cornish, op. cit., Ch. 11, pp. 345-346 (note 29).

44. Solar Thomson Engineering Co. v. Barton (1977) RPC 537 CA; see also Laddie et al., op. cit., pp. 48-50, paras. 2.75-2.76 (note 15); Cornish, ibid., Ch. 11, pp. 346-347 (note 29).

45. Laddie et al., ibid., (note 15); Cornish, ibid. (note 29).

46. John Cartwright, "How to identify an Infringer?", op. cit. (note 5).

47. Laddie et al., op. cit., p. 50, para. 2.76 (note 15); Cornish, op. cit., Ch. 11, pp. 345-346 (note 29).

48. John Cartwright, "How to identify the Infringer?", op. cit. (note 5).

49. Ibid.

50. Ibid.

51. Copyright Act, 1956, op. cit., Section 17(1) (note 4).

52. John Cartwright, "How to identify the Infringer?", op. cit. (note 5).

53. Reproduced in C.R. Franz, S.J. Wilkins and J.C. Bower, "A Critical Review of Proprietary Software Protection" in Information and Management, Vol. 4, No. 2, May 1981, p. 55 at p. 57.

54. Bryan Niblett, "Recent Copyright Developments in the United States", a presentation given on 12th June 1981 at a seminar entitled "Software Protection: The Current Issues" held at the London Press Centre.

55. Joseph P. Taphorn, op. cit., p. 33 (note 2); CONTU Report, op. cit., pp. 1-5 (note 2).

56. Copyright Act, 1909, USC Title 17, Section 4.

57. Communications of the ACM Briefs, Vol. 7, No. 7, July 1964, p. 450.

58. Ibid. (note 57).

59. Reed C. Lawlor, "Copyright and Patent Protection for Computer Programs", the Diebold Research Program - Professional Paper Series, 19.2.68-1.3.68, p. 2.

60. Reed C. Lawlor, "Copyright Aspects of Computer Usage", in Communications of the ACM 1964, Vol. 7, p. 572 at p. 575.

61. Copyright Act, 1976, USC Title 17, Section 102(a).

62. Ibid., Section 102(a)(1) (note 61); Niblett, Legal Protection of Computer Programs, op. cit., Ch. 7, p. 98 (note 2).

63. Definition reproduced in Announcement from the Copyright Office ML-252, February 1981, "Copyright Law Amended regarding Computer Programs", Public Law 96-517, 12th December 1980.

64. L.S. Nixon, op. cit., p. 351 (note 3).

65. Williams Electronic v. Artic International 685 F.2d. 870 (1982), reported in Copyright Law Reports, Vol. 51, 9/82, p. 17517 at p. 17517.

66. Interview with Duncan Davidson, chairman of the American Bar Association's Software Protection Sub-committee, 24th November 1982.

67. Copyright Act, 1976, op. cit., Section 106 (note 61).

68. Copyright Act, 1976, ibid., Section 101 (note 61).

69. Copyright Act, 1976, ibid., Section 101 (note 61).

70. GCA Corp. v. Chance, BNA's Patent, Trademark and Copyright Journal, Vo. 25, 11.11.82, Briefs p. 39.

71. Williams Electronic v. Artic International, op. cit., p. 17524 (note 65).

72. Ibid.

73. Tandy Corp. v. Personal Micro Computers Inc. 524 F.Supp. 171 at p. 173.

74. Copyright Act, 1976, op. cit., Section 101 (note 69).

75. Walter Klasson, "Copyright, Computers and the Betamax Case" in Byte, May 1982, p. 22 at p. 24, 28;
Sean Hallahan, "A Long Search for Effective Deterrents" in Computing, 24th June 1982, p. 23.

REFERENCES TO CHAPTER 3

1. Patents Act, 1977, Ch. 37, Section 25(1), p. 22.

2. Ibid., Section 60(1) (note 1).

3. Ibid., Section 60(1) (note 1).

4. Patents Act, 1949, 12, 13 and 14 Geo. 6, Ch. 87.

5. Patents Act, 1949, ibid., Section 101, p. 70 (note 4).

6. Patents Act, 1977, op. cit., Section 1(1), p. 1 (note 1).

7. Patents Act, 1952, 35 United States Code (USC).

8. Patents Act, 1952, ibid., 35 USC, Section 101 (note 7).

9. Patents Act, 1977, op. cit., Section 1(2), p. 2 (note 1).

10. Interview with R.J. Hart, Regional Patents Manager, Plessey Telecommunications Ltd., 29th May 1981.

11. Rolls Royce's Application 1963 Reports of Patent Cases (RPC) p. 251.

12. Patents Act, 1949, op. cit., Section 101, p. 70 (note 5).

13. NRDC's Application (1961) RPC No. 6, p. 135.

14. Ibid., p. 145, lines 47-49 (note 13).

15. Slee and Harris' Applications (1966) RPC No. 9, p. 194.

16. Ibid., p. 194, lines 1-5 (note 15).

17. Ibid., p. 196, line 48 - p. 197, line 7 (note 15).

18. Ibid., p. 197, lines 37-48 (note 15).

19. Ibid., p. 198, lines 22-26 (note 15).

20. Ibid., p. 198, lines 8-11 (note 15).

21. "Patentability of Computer Programs", IPD 38411, Industrial Property Department, Board of Trade, 1965.

22. Badger's Application (1970) RPC p. 36 at p. 37. line 16 - p. 38, line 4.

23. Ibid., p. 36, lines 5-9 (note 21).

24. Ibid., p. 38, lines 22-28 (note 21).

25. Ibid., p. 40, lines 14-34 (note 21).

26. Official Journal (Patents), 5th March 1969, p. 683.

27. Manual of Patent Office Practice (1972), para. 101, 23, 1.

28. Burroughs Corporation (Perkins') Application (1974) RPC p. 147 at p. 148, lines 26-36.

29. Ibid., p. 148, lines 39-40 (note 28).

30. Ibid., p. 153, lines 25-34 (note 28).

31. Ibid., p. 155, lines 10-19 (note 28).

32. Ibid., p. 155, lines 20-27 (note 28).

33. Ibid., p. 160, lines 15-20 (note 28).

34. Ibid., p. 160, lines 38-41 (note 28).

35. Manual of Patent Office Practice (1978), para. 101, 23, 1.

36. IBM's Application (1980) Fleet Street Law Reports (FSR) p. 564 at p. 573.

37. Patents Act, 1977, op. cit., Section 3, p. 4 (note 1).

38. Ibid., Section 1, p. 1 (note 1).

39. Ibid., Section 1(2), p. 2 (note 1).

40. Ibid., Section 1(2), p. 2 (note 1).

41. Convention on the Grant of European Patents (European Patent Convention, EPC), Treaty Series No. 20 (1978), Cmnd. 7090.

42. Patents Act, 1977, op. cit., Section 130(7), p. 110 (note 1).

43. "Guidelines for the Examination of European Patents", European Patent Office, Munich (Sept. 1980), Ch. IV, Part C, p. 25.

44. Interview with R.J. Hart, op. cit. (note 10).

45. "Software Protection: The Current Issues". A seminar held at the London Press Centre, 12th June 1981.

46. Manual of Patent Office Practice, Ch. 2100, Section 2110, October 1981.

47. Patents Act, 1952, op. cit., 35 USC, Section 101 (note 7).

48. Ibid., 35 USC, Section 100(b) (note 7).

49. Reproduced in "Statutory or Non-statutory: An Analysis of the Patentability of Computer Related Inventions" by D.A. Blumental and B.D. Riter in Journal of the Patent Office Society, August 1980, Vol. 62, No. 8, pp. 456-457.

50. Ibid., pp. 456-457 (note 49).

51. Ibid., p. 455 (note 49).

52. Ibid., p. 455 (note 49).

53. US Patent and Trademark Office Guidelines to the Examination of Programs, 829 Off. Gaz. Pat. Office 865 (16th August 1966), p. 865; see also Computer/Law Journal, Vol. 1, No. 1, Spring 1978, Appendix pp. 193-200.

54. D.A. Blumental and B.D. Riter, "Statutory or Non-statutory: An Analysis of the Patentability of Computer Related Inventions", op. cit., pp. 457-459 (note 49).

55. In re. Musgrave (431 F.2d. 882; 167 USPQ 280; CCPA 1970), comment reproduced in Blumental and Riter article, op. cit., p. 460 (note 49).

56. "Legal Protection for Computer Programs", Susan Hubbell Nyam, Computer/Law Journal, Vol. 1, No. 1, Spring 1978, pp. 43-44; Blumental and Riter article, op. cit., pp. 462-463 (note 49).

57. Gottschalk v. Benson, 34 L.Ed. 273 at p. 277, the Court citing dicta from Funk Bros. Seed Co. v. Kalo C. 333 US 127, 130, 92 L.Ed. 588, 68 S. Ct. 440.

58. Gottschalk v. Benson, ibid., p. 279 (note 57).

59. Parker v. Flook 57. L.Ed. 2d. 451 at pp. 454-455.

60. Parker v. Flook, ibid., p. 455 (note 59).

61. Parker v. Flook, ibid., p. 455 (note 59).

62. Parker v. Flook, ibid., p. 456 (note 59).

63. Parker v. Flook, ibid., p. 455 (note 59).

64. Parker v. Flook, ibid., p. 451 (note 59).

65. F.J. Jordan, "Computer Related Invention held Patentable by US Supreme Court" in Patents and Licensing, June 1981, Vol. XI, No. 3, p. 28.

66. Diamond v. Diehr 67 L.Ed. 2d., No. 2, p. 155 at pp. 155-156.

67. Diamond v. Diehr, ibid., p. 169 (note 66).

68. Diamond v. Diehr, ibid., p. 158 (note 66).

69. Manual of Patent Office Practice, Chapter 2100, Section 2110, October 1981, op. cit. (note 46).

70. L.S. Nixon, "Protection of Intellectual Property Rights in the United States for Computer-aided Processes, Machines and Audio-visual Displays" in The Journal of the Chartered Institute of Patent Agents, Vol. 12, No. 8, May 1983, p. 347.

71. Manual of Patent Office Practice, op. cit., pp. 4-5 (note 46).

72. Manual of Patent Office Practice, ibid., p. 5 (note 46).

73. L.S. Nixon, op. cit., p. 347 (note 70).

74. Gareth Jones, "Restitution of Benefits obtained in Breach of Another's Confidence", 1970, Law Quarterly Review 86, p. 465 and The Law Commission, Working Paper No. 58, "Breach of Confidence", HMSO, 1974, para. 16, p. 11.

75. Gareth Jones, ibid. (note 74); Law Commission Working Paper no. 58, ibid. (note 74).

76. Fraser v. Evans (1969) I Q.B. p. 349 at p. 361; Gareth Jones, ibid., p. 466 (note 74); Law Commission Working Paper No. 58, ibid., para. 16, p. 11 (note 74); Colin Tapper, Computer Law, Longmans, 1978, Ch. 1, p. 30.

77. Law Commission Working Paper No. 58, ibid., para. 6, pp. 3-4 (note 74).

78. Gareth Jones, op. cit., p. 463 (note 74).

79. William Cornish, Intellectual Property: Patents, Copyright, Trademarks and Allied Rights, Sweet and Maxwell, 1981, Ch. 8, pp. 263-264.

80. Gareth Jones, op. cit., p. 464 (note 74).

81. Colin Tapper, op. cit., Ch. 1, p. 22 (note 76).

82. Saltman (Engineering) Co., Ferotec Ltd. and Monarch Engineering Co. (Mitcham) Ltd. v. Campbell Engineering Co. Ltd. (1948) 65 RPC 203 at p. 215.

83. Coco v. A.N. Clark (Engineers) Ltd. (1969) RPC No. 2, at p. 47.

84. William Cornish, op. cit., p. 268 (note 79).

85. Saltman (Engineering) Co. et al. v. Campbell Engineering Co. Ltd., op. cit., at p. 215 (note 82).

86. Coco v. A.N. Clark, op. cit., p. 47 (note 83).

87. Coco v. A.N. Clark, ibid., p. 48 (note 83).

88. William Cornish, op. cit., p. 268 (note 79).

89. William Cornish, ibid., p. 274 (note 79).

90. Morison v. Moat (1851) 9 Hare 241; Law Commission Working Paper No. 58, op. cit., para. 7, pp. 4-5 (note 74).

91. Prince Albert v. Strange (1849) Mac and G. 25; Law Commission Working Paper No. 58, ibid., para. 7, pp. 4-5 (note 74).

92. Triplex Safety Glass Co. v. Scorah (1938) 55 RPC 21; Reid and Sigrist Ltd. v. Moss Mechanism Ltd. (1932) 49 RPC 461.

93. William Cornish, "Protection of Confidential Information in English Law" in IIC (1975) Vol. 6, No. 1, p. 50-53.

94. William Cornish IIC article, ibid., p. 50-53 (note 93).

95. Bryan Niblett, Legal Protection of Computer Programs, Oyez, 1980, pp. 71-72.

96. Bryan Niblett, ibid., p. 71-72 (note 95).

97. Bryan Niblett, ibid., p. 71-72 (note 95).

98. Colin Tapper, op. cit., p. 34 (note 76), (Hirac Ltd. v. Park Royal Scientific Instruments Ltd. 1946, Ch. 169).

99. William Cornish, op. cit., p. 275 (note 79), (Bents Brewery v. Hogan 1945 2 All E.R. 570).

100. William Cornish, ibid., p. 275 (note 79), (Printers and Finishers Ltd. v. Holloway 1965 RPC 239).

101. William Cornish, ibid., p. 275 (note 79), (Robb v. Green 1895 2 Q.B. 315).

102. William Cornish, ibid., p. 275 (note 79).

103. Bryan Niblett, op. cit., p. 71-72 (note 95), (Amber Size Co Ltd. v. Herzel 1913 2, Ch. 239).

104. William Cornish, op. cit., p. 175 (note 79).

105. Bryan Niblett, op. cit., p. 71-72 (note 95), (Nordenfelt v. Maxim Nordenfelt Guns and Ammunition G. 1894 AC 535).

106. Bryan Niblett, ibid., p. 71-72 (note 95).

107. Colin Tapper, op. cit., p. 34 (note 76).

108. Law Commission Working Paper No. 58, op. cit., para. 25, pp. 23-25 (note 74).

109. William Cornish, op. cit., pp. 283-284 (note 79).

110. Law Commission Working Paper, op. cit., paras. 33-37, pp. 31-34 (note 74).

REFERENCES TO CHAPTER 4

1. Margaret L.B. Anderson, Software Protection: A Survey of the Industry, Final Report, University of Kent, August 1976.

2. Richard I. Miller, "The CONTU Software Protection Survey" in Jurimetrics Journal, Summer 1978, Vol. 18, No. 4, p. 354.

3. Alan Cane, "Computer Services and Software", in Computing Services and Software IV, Financial Times, 5th November 1982.

4. Peter Large, "Wonder Chip takes up Less Space", The Guardian, 17th November 1982, p. 15.

5. "Which Computer?", Business Computer Guide, March 1982, p. 107.

6. Richard I. Miller, op. cit., pp. 365-366 (note 2).

REFERENCES TO CHAPTER 5

1. S.T. Kent, "Protecting externally supplied Software in Small Computers", MIT/LCS-TR-255, Sept. 1980, p. 17.

2. Ibid., p. 14 (note 1).

3. B.J. Walker and I.F. Blake, Computer Security and Protection Structures , 1st Edition, Dowden, Hutchinson and Ross Inc., 1977, ISBN: 0-87933-247-6, p. 21.

4. S.T. Kent, op. cit., pp. 97-98 (note 1); D.W. Davies and D.A. Bell, "The Protection of Data by Cryptography", NPL Report Com. 98, Jan. 1978.

5. Dr. Donald Davies, "Recent Developments in Cryptography", a talk given to the Technology of Software Protection Group of the British Computer Society on 17th May 1982.

6. S.T. Kent, op. cit. (note 1).

7. Ibid., pp. 28-33 (note 1).

8. Ibid., p. 27 (note 1).

9. Ibid., pp. 28-31 (note 1).

10. Ibid., pp. 32-33 (note 1).

11. Interview with John Cartwright, Group Patent Services, ICL, 14th May 1981.

12. SOURCE CONFIDENTIAL.

13. Interview with Nigel Parry, Product Marketing Manager, Apple Computer (UK) Limited, 3rd August 1982.

14. Interview with Mike Whitehead, Bristol Software Factory, 25th January 1983.

15. Interview with Duncan Davidson, Chairman of the Subcommittee on Software Protection, American Bar Association, 24th November 1982.

16. Interview with Roy Stephenson, Claremont Controls, 19th January 1983.

17. Interview with Mike Lake, Wordcraft Systems, 3rd February 1983.

18. Interview with Ritchie McGladdery, File Tab Support Services 15th June 1982; "Is there a Time Bomb in your Machine?" in New Scientist, 28th October 1982; interview with David Keogh, Dataskil, 2nd June 1981.

19. Interview with David Reeve, Grade One Computing Services Limited, 14th May 1982.

20. Interview with Ian Monro, Managing Director, Computerised Business Systems Limited, 6th August 1982.

21. "CBM Disk Protection", report by John Collins, Commodore Business Machines (UK) Ltd., 24th February 1981; Derrick Grover, "The Technology of Software Protection" in Computer Bulletin, Series 2, March 1981, No. 27.

22. Mike Lake, op. cit. (note 17); Paul Handover, Dataview Ltd., "A Distributor's View of Piracy and Methods of Prevention", a talk given to the Technology of Software Protection Specialist Group of the British Computer Society, 14th July 1981.

23. Interview with Neil Cornish, Sales Manager, TABS Limited, 3rd June 1982.

24. Interview with A.J. Swan, General Manager, Vlasak Computer Systems, 28th September 1982.

25. Interview with Sid Newman, Dataview Limited, 2nd March 1982; Mike Whitehead, op. cit. (note 14); Mike Lake, op. cit. (note 17).

26. Mike Whitehead, ibid. (note 14).

27. Mike Lake, op. cit. (note 17).

28. Mike Lake, ibid. (note 17).

29. Mike Lake, ibid. (note 17).

30. A.J. Swan, op. cit. (note 24); Mike Lake, ibid. (note 17); Sid Newman, op. cit. (note 25); Mike Whitehead, op. cit. (note 14).

31. Mike Lake, ibid. (note 17).

32. Interview with Sandy Livingstone, Technical Manager, Taylor-Wilson Systems Limited, 17th September 1982.

33. Interview with Shane Barnes, MMS Software Limited, 26th August 1982.

34. A.J. Swan, op. cit. (note 24).

35. Neil Cornish, TABS Ltd., "The Practicalities of Hardware Protection: TABS Experience", a talk given to the Technology of Software Protection Specialist Group of the British Computer Society, 25th April 1983; Sandy Livingstone, op. cit. (note 32).

36. A.J. Swan, op. cit. (note 24).

37. Mike Whitehead, ibid. (note 14).

38. Interview with Peter Miller, Microprocessor Unit, University of Aston, 8th February 1983; A. Lesea and R. Zaks, Microprocessor Interfacing Techniques, Sybex Inc., 1978, ISBN: 0-89588-003-2, Appendix D, p. 405.

39. Interview with Neil Cornish, op. cit. (note 23).

40. Interview with John Chew, Kingston Computers Limited, September 1982; Sandy Livingstone, op. cit. (note 32).

41. Sandy Livingstone, ibid. (note 32).

42. Mike Lake, op. cit. (note 17); Neil Cornish, interview, op. cit. (note 23); Ian Monro, op. cit. (note 20); Shane Barnes, op. cit. (note 33); A.J. Swan, op. cit. (note 24).

43. Interview with David Hughes, Drive Technology Limited, 1st February 1983.

44. Interviews with John Baldachin, Little Genius Limited, 27th July 1982; Raj Narayanan and Michelle Boeglin, SoftGuard Computer Systems, 5th August 1982; and John Gamson, Microhex Computers, 14th September 1982; 'APPLEGUARD', Sensible Software Product Literature (January 1982).

45. Claire Gooding, "Helping to Lock the Door against CP/M Bootleggers", Computer Weekly, 26th February 1981, p. 8; SoftGuard Computer Systems Product Literature (August 1982); 'APPLEGUARD' literature, Sensible Software Inc., ibid. (note 44); John Collins, "CBM Disk Protection", op. cit. (note 21).

46. Louis E. Frenzel, Getting Acquainted with Microcomputers, Howard W. Sams and Co. Inc. 1978, ISBN: 0-627-21486-5, pp. 125-6.

47. Lesea and Zaks, op. cit., pp. 152-256 (note 38).

48. John Gamson, op. cit. (note 44).

49. "Battle against Software Piracy begins", New Scientist, 1st October 1981; Nigel Parry, op. cit. (note 13).

50. Mike Gurr, "Some Technical Aspects of the Protection of Software", a presentation at Gower Software Protection Conference, 3rd March 1981, Waldorf Hotel, London; Mike Lake, op. cit. (note 17).

51. Nigel Parry, op. cit. (note 13).

52. Mike Gurr, presentation, op. cit. (note 50).

53. David Hughes, op. cit. (note 43).

54. Mike Whitehead, op. cit. (note 14).

55. David Hughes, op. cit. (note 43).

56. Software File 2, Computer Weekly, 12th March 1981, p. 9; David Hughes, ibid. (note 43).

57. John Baldachin, op. cit. (note 44).

58. Mike Whitehead, op. cit. (note 14); Mike Lake, op. cit. (note 17); Raj Narayanan and Michelle Boeglin, op. cit. (note 44).

59. Louis E. Frenzel, op. cit., pp. 44-6 (note 46).

60. Mike Whitehead, op. cit. (note 14); Raj Narayanan and Michelle Boeglin, op. cit. (note 44).

61. Louis E. Frenzel, op. cit., pp. 172-3 (note 46).

62. Ibid., pp. 130-137 (note 46).

63. Mike Lake, op. cit. (note 17).

64. Raj Narayanan and Michelle Boeglin, op. cit. (note 44); John Collins, "CBM Disk Protection", op. cit. (note 21).

65. Raj Narayanan and Michelle Boeglin, ibid. (note 44).

66. David Hughes, op. cit. (note 43).

67. John Gamson, op. cit. (note 44).

68. Ian Monro, op. cit. (note 20).

69. Maggie Mclening, "Piracy killing off New Product Launches", in Computer Weekly, Software File, 26th August 1982, p. 5.

70. Interview with Derrick Grover and Ken Cunningham, British Technology Group, 18th November 1981.

71. Interviews with Tony Plackowski, Tridata Micros Limited, 6th November 1981; E. Ric Giardina, Legal Counsel, MicroPro International, 17th November 1982; and Gerry Davis, Legal Counsel, Digital Research Inc., 22nd November 1982.

72. A.J. Swan, op. cit. (note 24).

73. Interview with Laszlo Belady, Program Manager, Software Technology Corporate Headquarters, IBM, Armonk N.Y., 30th November 1982.

74. David Reeve, op. cit. (note 19).

75. Interview with Peter Laurie, Southdata Limited, 12th March 1982; E. Ric Giardina, op. cit. (note 71).

76. Interview with David Sheppard, Marketing Manager, Datacall Limited, 2nd September 1982.

77. Laszlo Belady, op. cit. (note 73); David Reeve, op. cit. (note 19).

78. David Hughes, op. cit. (note 43); Mike Lake, op. cit. (note 17).

REFERENCES TO CHAPTER 6

1. Interview with Peter Laurie, Managing Director of Southdata Ltd., and Editor of Practical Computing, 12th March 1982.

2. Interview with David Hughes, Drive Technology Ltd., 1st February 1983.

3. Interview with E. Ric Giardina, Legal Counsel, MicroPro International Inc., 17th November 1982.

4. Interview with Duncan Davidson, Chairman of the American Bar Association's Software Protection Sub-committee, 24th November 1982.

5. Bryan Niblett, Legal Protection of Computer Programs, Oyez Publishing Ltd., London, 1980, 'Coda' p. 111.

6. Interview with John Cartwright, Manager, Group Patent Services, ICL, 14th May 1981.

7. Interview with Sid Newman, Dataview Ltd., 2nd March 1982; SoftGuard Computer Systems product literature, August 1982.

8. Simon Elsom, "Computer Software Protection: A Survey of the Industry", Proceedings of the Technology of Software Protection Group Meeting, British Computer Society, 14th March 1983.

9. British Computer Society Copyright Committee document CC/1/83, p. 1.

10. Ibid., p. 1 (note 9).

11. Ibid., p. 2 (note 9).

12. Ibid., p. 1 (note 9).

13. Ibid., p. 2 (note 9).