UNIVERSITY OF SOUTHAMPTON

ANALYSIS OF THE LEGALITY OF "REVERSE ENGINEERING" OF COMPUTER PROGRAMS UNDER THE COPYRIGHT DESIGNS AND PATENTS ACT 1988: AN APPROACH FOR THE FUTURE

LT. TANAPHOT EKKAYOKKAYA

LL.B., LL.M. (Hons), Solicitor & Barrister

Ph.D. THESIS

FACULTY OF LAW

MAY 2001

DEDICATION

This thesis is dedicated to my family – mother and father, Keith and Marian, grandma, Aunt Sumana, Manapol, and Pollarat – and to the Thai Government.

This thesis is the result of work done wholly while the author was in registered postgraduate candidature.

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF LAW

Doctor of Philosophy

ANALYSIS OF THE LEGALITY OF "REVERSE ENGINEERING" OF COMPUTER PROGRAMS UNDER THE COPYRIGHT DESIGNS AND PATENTS ACT 1988: AN APPROACH FOR THE FUTURE

by Lt. Tanaphot Ekkayokkaya

This thesis analyses the legality of "reverse engineering" under the Copyright Designs and Patents Act 1988 and suggests a new way to consider the legality of reverse engineering. The thesis submits that current copyright law misconceives the principle of software engineering and, as a result, fails to differentiate the process of reverse engineering from that of forward engineering. Such a conceptual misunderstanding creates commercial problems in the software industries and other industries which rely on the use of software technology.

As a means of solving the problems, this thesis suggests that these two processes should be separated from each other so that the legal status of each one is considered on its own merits. The thesis proposes that "reverse engineering", and the steps leading to the completion of a finished product, i.e., "forward engineering", should not be an infringement of copyright. Rather, infringement should be determined by a comparison of the finished product with the original product. Comparison of the two products will create a more open creative environment for the exploitation of ideas, and stimulate greater encouragement of competition in the software market.

Also examined is the impact of the framework proposed in this thesis on three main related areas of law, namely, the law of confidence, patent law and competition law. The thesis shows that the proposed framework will not have an adverse impact on these areas. The author believes that this thesis is the first to introduce a new way of perceiving and solving the problems of the legality of reverse engineering as well as analysing the impact of the proposed framework on related areas of law.

TABLE OF CONTENTS

PREFACE

The debate on software reverse engineering has been going on for more than a decade. Much of the discussion has focused on the extent to which reverse engineering should be legally allowed without considering the actual process of software engineering in which reverse engineering is embedded. Therefore, the legal solution to date simply complicates the matter and interferes with the doctrine of copyright law. In the United Kingdom and the EC, the legal solution is not only ambiguous but also inefficient when solving the commercial problems. Furthermore, the legal solution which was propounded in the early 1990s, it is submitted, cannot cope with the swift development of software technology. The development in this field over the past five years gives rise to the need for the legality of reverse engineering beyond that which was provided in the early 1990s. Thus, it can be said that the Copyright Designs and Patents Act 1988 is one step behind the current development of software technology and is waiting for reconsideration.

These are the circumstances which have brought about my research.

I argue that the stringent conditions of the reverse engineering privilege provided in the Software Directive and the Copyright Designs and Patents Act 1988 fail to accommodate a broad range of real-life situations. They also fail to meet the objective of supporting the standardisation in the software industry. I suggest that a relaxation on reverse engineering would have better promoted the software industry's desire to break into the international market, thus facilitating the creation of interoperable and open systems.

At present, it seems that neither the EC Directive nor the Copyright Designs and Patents Act 1988 can serve as the best model for achieving the standardisation in the software market or promoting the ability of the EC software industry to compete with the US counterpart. In addition, this stringent decompilation privilege reflects the short-sightedness of both the EC and SAGE; as one scholar expresses her concern by stating that "it injures not only the developing EC software industry but someday may backfire and hinder current industry giants ..., for if and when another competitor,

such as the Japanese software industry, develops a product that becomes the industry standard, who then will be begging for the economically sensible right of reverse engineering to better promote competition and innovation?".[1] It is, therefore, the aim of this thesis to represent a broader and clearer reverse engineering privilege. This recommended privilege would bring the software industry of all members of the EC into line with global standardisation and perform as a model for other countries in carrying out the open systems.

The aim of this thesis can be broken down into following objectives:

(i) To provide a recommendation for a legal framework which would give a broader scope for the legality of reverse engineering and which could be incorporated into an EC Directive or enacted as an amendment to the Copyright Designs and Patents Act 1988, or, at the very least, considered as a foundation for a prospective discussion at both national and international levels.

(ii) To provide a comprehensive and critical study of the principle of copyright law governing reverse engineering computer programs; to examine the policies of the EC in introducing the Directive on the legal protection of computer programs (91/250/EEC), 1991 O.J. (L122); to scrutinise the policy objectives of the UK Government in the adoption of the Software Directive by inserting section 50B into the Copyright Designs and Patents Act 1988; to examine whether such incorporation of the contents of section 50B provides the same effect as that of Article 6 of the Software Directive; to assess whether Article 6 and section 50B, or, broadly speaking, the Software Directive and the Copyright Designs and Patents Act 1988 can satisfy the need of reverse engineering in current software markets.

(iii) To trace the development of case law in the United States with respect to reverse engineering; to identify the parameters of the legality of reverse engineering in the United States; to signify the trend of the US courts in legalising reverse engineering to comply with "open systems" in future; to suggest, in accordance with (i), the trend of

---

[1] Morrison G Linda, 'The EC Directive on the Legal Protection of Computer Programs: Does It Leave Room for Reverse Engineering Beyond the Need for Interoperability?' (1992) 25 Vanderbilt Journal of Transnational Law 293, at 332.

the legality of reverse engineering in the United Kingdom so as to fulfil the global need of the open systems.

(iv) To analyse the relationship between reverse engineering and copyright law in terms of what aspects of the use of the reverse engineering technique could infringe the exclusive rights of the copyright owner; to trace the development of the idea/expression dichotomy and analyse its relationship with reverse engineering; to examine the necessity of using reverse engineering techniques in reaching the standardisation of user and technical interface.

(v) To analyse the impact of the proposed framework on three main related areas of law, namely, the law of confidence, patent law and competition law.

To accomplish these purposes, first of all, the thesis will provide for a basic understanding of the common practice and procedure of reverse engineering that is necessary for the discussion in greater detail in the following chapters. This includes the explanation of the definition of reverse engineering. The practices of reverse engineering in software development and software maintenance are separately considered. This requires consultation with the staff and review of the literature from the Engineering Department.

Secondly, reverse engineering under copyright law in the EC will be examined and compared to reverse engineering under the Copyright Designs and Patents Act 1988, which is based on the EC Directive, so as to clarify whether or not it provides sufficient provisions to balance the need of the protection of computer programs and the retention of competitiveness in software markets. In this analysis, cases relating to reverse engineering both occurring before and after the enactment of the Copyright Designs and Patents Act 1988 will be canvassed. In order to understand the basis of the Copyright Designs and Patents Act 1988, the EC Software Directive will be analysed. The Internet is of great assistance in searching for related information. For a comparative analysis, copyright law in the United States will be chosen as it well represents the use of the common law doctrine of "fair use" which the US courts employ to legalise reverse engineering.

Finally, this thesis will examine the impact of the proposed framework on the law of confidence, patent law and competition law. In particular, it will compare the possible use of competition law to legalise reverse engineering with the proposed framework. The impact of the proposed framework is limited to only two branches of intellectual property law, namely patent law and the law of confidence,[2] because they provide rather direct legal protection for computer programs, as opposed to trademark and designs law. A knowledge of economics needs to be applied in order to analyse the impact on these areas of law and to justify the proposed framework.

This author has attempted to state the law as it stands on 1 April 2001.

Tanaphot Ekkayokkaya
Faculty of Law
University of Southampton

[2] The law of confidence can be regarded as an intellectual property right to the extent that equity enables its owner to control the dissemination of the information to others and, in certain circumstances, provides a remedy in the event of its unauthorised disclosure. See Stephen Saxby, *Encyclopedia of Information Technology Law* Volume I (Sweet & Maxwell: London, 1990-2000) para. 2.2000.

ACKNOWLEDGEMENTS

In writing this thesis, I received help from many people. First of all, I would like to thank my supervisor, Dr. Stephen Saxby, who helped immensely during the arduous process. I also would like to thank Professor Nicholas Gaskell for helping me to draft the proposal of my thesis, and Professor Charles Debattista and Dr. Andrew Halpin for their invaluable suggestions.

My studies in Southampton would not have been enjoyable without great encouragement and support from my host family, Dr. Keith & Marian Hubble. I very much appreciate their hospitality.

I am very much grateful for financial support from the Thai Government and the Faculty of Law, University of Southampton.

Thanks, also, to my friends – Wachira Boonyanet, Sita Sitalux and other Thai students who gave me encouragement during my studies. Also many thanks to my officemates – Stuart Macdonald who gave me a piece of cake and Kyriaki Noussia who helps me put on weight by donating loads of Greek chocolate.

And last, but by no means least, thank you mum and dad for everything.

DEFINITIONS AND ABBREVIATIONS

1. "Applets" – Java programs that run across the Internet.
2. "Black box" reverse engineering – a method of observing the functioning of a program without access to the source code.
3. "Byte code" – An intermediate form between source code and object code.
4. "CAD" – Computer-aided design. CAD is a technology that is widely used in science and engineering. Using CAD systems, engineers may design new products, invent new processes, or even create other software programs that are based on interactions between their own experience and expertise with a CAD system.
5. "CAM" – Computer Aided Manufacturing. A combination of CAD and CAM enables a designer to create a three-dimensional representation of an object, with the help of the computer, and then the computer programs instructions for automated manufacture of the object and controls the manufacturing process.
6. "CDPA 1988" – Copyright Designs and Patents Act 1988.
7. "Comment" – text in a program that is not meant for seeing by the user but is meant for a statement so that the programmer or someone looking at the program can know what is going on.
8. "CONTU" – The National Commission on New Technological Uses of Copyrighted Works.
9. "CUE" – Computer Users in Europe.
10. "Decompilation" – the other type of reverse engineering which requires a conversion of object code towards source code and related data. (In both "black box" reverse engineering and "decompilation", the target program needs to be run in a computer and thus a copy of the target program will be created in RAM. A copy that is created in RAM, although a temporary form of expression, will constitute an infringement of copyright, unless it is authorised by the copyright holder. In addition, in the case of decompilation, the conversion of object code to source code is also considered to be adaptation, which is one of the exclusive rights of copyright. For this reason, the copyright holder can assert his exclusive rights to prevent others from accessing ideas in his computer program.)
11. "ECIS" – The European Committee for Interoperable Systems.

12. "EPC" – European Patent Convention 1973.
13. "EPO" – European Patent Office.
14. "Forward engineering" – the process of rebuilding a new application system, using the ideas, objects, etc. retrieved from "reverse engineering". (The process of reverse engineering inherently shifts into the process of forward engineering once the reverse engineering process is completed at some point.)
15. "GATT" – General Agreement on Tariffs and Trade.
16. "IP" – Intellectual Property.
17. "IT" – Information Technology.
18. "Legacy system" – A computer system or application program which continues to be used because of the cost of replacing or redesigning it and often despite its poor competitiveness and compatibility with modern equivalents. The implication is that the system is large, monolithic and difficult to modify. If legacy software only runs on antiquate hardware the cost of maintaining this may eventually outweigh the cost of replacing both the software and hardware unless some form of emulation or backward compatibility allows the software to run on new hardware.
19. "PCT" – Patent Co-operation Treaty.
20. "PLT" – Patent Law Treaty.
21. "PO" – Patent Office (normally refer to UK Patent Office).
22. "RAM" – Random Access Memory.
23. "Reverse engineering" – the process of analysing how a computer program functions. Reverse engineering can be performed in two different ways: by "black box" and "decompilation" methods.
24. "ROM" – Read Only Memory.
25. "SAGE" – The Software Action Group for Europe.
26. "SMEs" – Small and Medium sized Enterprises.
27. "Software engineering" – a generic term covering both the processes of "reverse engineering" and "forward engineering".
28. "SRI" – software related invention.
29. "TRIPs" – Trade Related Aspects of Intellectual Property Rights.
30. "WIPO" – World Intellectual Property Organisation.
31. "WTO" – World Trade Organisation.
32. "Y2K" – The year 2000 problem.

CHAPTER ONE

INTRODUCTION

---

1. **Overview of the software industry**
   1.1 Overview the recent development of software
   1.2 Trend of software development in future
2. **The importance of reverse engineering**
3. **The concept of intellectual property law**
   3.1 IP law and its role in protecting commercial interests
   3.2 IP law and its role in benefiting society
4 **The issues**
5 **The thesis**
   5.1 The thrust of the thesis
   5.2 Testing the proposed framework against the philosophy underlying IP law
   5.3 The impact of the proposed framework on other related areas of law
6 **The structure of the thesis**

1. OVERVIEW OF THE SOFTWARE INDUSTRY

1.1 Overview the recent development of software

In recent years, computer programs have become almost indispensable in many industries and organisations. This has been evidenced by the fact that many products and services on the market have made use of computer technology as a selling point. Nowadays, as the use of computer technology is even increasing, most companies are left with only two choices, namely, adopt the technology and stay in business, or ignore it and trail behind their competitors.

Such a move towards the computer technology-oriented sphere not only exists in the industry where computer programs are marketed as independently finished products but also in other industries where computer programs are embedded in or incorporated within other products. For instance, in the automobile and communication industries, computer programs are used extensively. This leads to an unprecedented achievement

which would never have happened without the aid of computer programs. By way of an example, many car companies now use computer programs to control brake pressure and the distribution of torque on each wheel to prevent the car from understeering or oversteering in the mid bend.[1] This system is called "traction control". For the communications industry, mobile phones appear to be the best example. Nowadays, many mobile phones contain voice-activated dialling software[2] and Wireless Application Protocol (WAP) enabling the user to search the Internet via mobile phones.[3]

This move towards the computer technology-oriented sphere is also held to be true for other industries where the efficiency of services is crucial. In the banking industry, for example, computer programs are used to enable customers to perform many banking transactions, e.g. transfer funds between their accounts, pay their bills and view their mortgage accounts over the Internet, without the restriction of banking hours.[4] Moreover, with the aid of the computer technology, not only can traditional commercial transactions[5] be carried out over the Internet but new breeds of businesses are also created, e.g. reverse auction and get-paid-to-surf-the-Internet businesses.[6]

It is, therefore, quite manifest that computer programs now have an important role to play in our "information society". Thus, their applications to modern businesses are not negligible.

---

[1] Car companies that have used this system are, for example, Mercedes, BMW, Volvo, Lexus, Audi, Jaguar, Ferrari and Porsche. Nathan Morgan, 'Road test new BMW 5-Series'/ Mercedes E-Class (2001) 634 Auto Express 34, 36; Nik Berg, 'Family planning – Volvo V70' (2000) 79 Top Gear 29; Peter Grunert, 'Middle class' (2000) 70 Top Gear 122, 123; Michael Bailie, 'Fully loaded' (2000) 70 Top Gear 154, 156-7; James Mills, 'Jag's new R-types' (2000) 573 Auto Express 26, 29. Tom Stewart, 'In bed with Modena' (1999) 69 Top Gear 96, 98; Richard Meaden, 'Porsche 911 Carrera 4' (1998) 2 EVO 68, 77.

[2] For example, Ericsson T18 and T28.

[3] For example, Nokia 7110 and 6210.

[4] For example, the Internet banking services offered by HSBC Bank and Lloyds TSB Bank, <http://www.banking.us.hsbc.com/internetbanking/faqs.asp>, <http://www.infoville.org.uk/banking/online/banking_lloydtsb.htm> respectively, accessed on 29/01/01. Companies that help creating Internet banking software are, for example, WoldNet Bank Inc. <http://www.worldnetbank.com>, Fiserv Solutions Inc. <http://www.fisnet.com> accessed on 29/01/01.

[5] For example, selling books or air tickets.

[6] The Get Paid companies earns their money on the advertisements they bring to the web-surfer and that is the reason why they can keep up paying the web-surfer. The more time the web-surfer spends

1.2 Trend of software development in future

As the growth of the application of software to various industries has increased at a rapid rate, it can be predicted with certainty that within a few years computer software will become a significant element for new kinds of technology-oriented innovations in many industries.

Consider two of the above examples. In the communications industry, a project, called the "Transparently Reconfigurable Ubiquitous Terminal" (TRUST), has just been commenced to create a seamless communication across the information society.[7] The project is set to create an imminent wave of the future which is seen as "mobile multimedia". This will, in the near future, replace the contemporary slogan of "anyone, anywhere, anytime", which refers to simple voice telephony, with the new slogan of "everything, everywhere, always on-line".[8] Such a technological breakthrough in this project relies heavily on the swift development of software. In this particular example, the project is achieved by software-reconfigurable radio systems and networks.[9] Similarly, in the automobile industry, tomorrow's cars in 2006 will fully packed with an Aladdin's cave of technology, e.g. Global Positioning Systems (GPS),[10] intelligent airbags,[11] steer-by-wire technology,[12] electro-hydraulic valves[13] and Sensotronic Brake Control (SBC).[14]

---

online the more money they will earn. Examples of those companies are destopdollars.com, cashsurfers.com, <http://www.get.paid.to.surf-the.web.homepage.com> accessed on 30/01/01.

[7] TRUST <http://www.ist-trust.org/trust_frameset.html> accessed on 31/02/01.

[8] Ibid.

[9] Ibid. Personal mobile multimedia based around UMTS has accelerated fast since the baseline air interface decisions of ETSI in January 1998, with 3rd Generation (3G) multimedia services and technologies beginning to take shape. The role of software re-configurable radio within this framework is beginning to clarify - within Europe the need to evolve GSM infrastructures to support UMTS and the need to roam between GSM and UMTS infrastructures will require flexible implementations. The TRUST programme is aimed at the development of these enabling technologies and associated Intellectual Property within Europe.

[10] Ian Adcock, 'Inside story: This is the future' (2001) 635 Auto Express 32, 34. It helps the driver to plot the route to avoid traffic jams and contacts the emergency services in case of accident.

[11] Ibid. The system using electronic sensors will control whether the airbag is to be deployed and at what rate to minimise injuries.

[12] Ibid. This technology will replace a "mechanical link", i.e. the steering column between the driver and the front wheels. The companies developing this technology are BMW and Delphi.

[13] Ibid., at 32. Instead of camshafts, rocker arms and tappets, tiny electric coils combined with hydraulics will operate each valve independently so that the cylinder receives the precise amount of fuel and air needed, whatever the driving conditions. This technology is being developed by Lotus Engineering.

As there will be a sharp increase in the use of software technology, it is expected that proficient communication between different kinds of software will become vital. For example, it will be necessary for an online banking program to be able to communicate with a personal financial program of a user at home.[15] It will also be necessary for the GPS to be compatible with the steer-by-wire technology in order to provide maximum safety. The Massachusetts Institute of Technology is working to link electronic steering and braking systems into the GPS navigation network, so that if a driver starts to deviate off the road the computer would automatically keep the vehicle on the straight.[16]

Therefore, it is anticipated that the development of software technology will lead to an immense increase in research and application of computer software to various industries and will result in the need for compatibility between different kinds of software. This compatibility will to a great extent be dependent on the development of reverse engineering.

2. THE IMPORTANCE OF REVERSE ENGINEERING

One of the most important processes in the development of software technology mentioned above is that of reverse engineering. Reverse engineering in the computer science language refers to the process of analysing a computer program. It can be said that reverse engineering plays an important role in both software development and software maintenance because the programmer or the engineer usually performs reverse engineering in the course of developing new software and maintaining existing software in order to retrieve necessary information. This information can then be used to achieve several goals, e.g. to ensure compatibility between computer programs, to create a competing program and to correct errors that have occurred during the lifecycle of a computer program. Full details of reverse engineering will

---

[14] Ibid., at 32. This optimises the braking effect via a micro processor so that each wheel has maximum stopping effort according to speed and road conditions.
[15] At the present time, only Microsoft Money is able to communicate with HSBC Bank's online banking services, supra note 4.
[16] Supra note 10, at 32.

be given in the next chapter. For reverse engineering to become widely acceptable, changes will be required in the law relating to intellectual property.

3. THE CONCEPT OF INTELLECTUAL PROPERTY LAW

The development of software technology can be said to be the product of genius and, thus, requires some form of legal protection. Undeniably, the most straightforward applicable legal framework is intellectual property law (IP law). This section briefly describes the role of IP law in protecting commercial interests and in benefiting society.

3.1 IP law and its role in protecting commercial interests

It has long been accepted that the most basic function of intellectual property law (IP law) is to provide a form of a legal protection for what a man has produced and brought into being.[17] In effect, IP law has provided a framework which ensures that the publication of new works and the manufacture of new products will be profitable, provided that they are sufficiently meritorious, useful and commercially attractive to attain a viable level of sales.[18] In case of computer programs, copyright law is the branch of IP law which has been accepted as a means of protecting computer programs. Therefore, the natural role of copyright law in this respect is to provide the author with legal protection for his commercial interests derived from his computer programs. Thus, it can be seen that, in this context, IP law functions as a property right which is granted as incentives and rewards. This leads to the following rights.[19]

1. Right to possess something,
2. Right to use or enjoy its benefits,
3. Right to manage or decide how it is to be used,
4. Right to receive income from it,

---

[17] David Bainbridge, *Intellectual Property 4th ed.* (FT Pitman Publishing: London, 1999) p. 18.
[18] Ibid.
[19] The Office of Technology Assessment (OTA), *Intellectual Property Rights in an Age of Electronics and Information* (1986), p. 21 quoting Lawrence Becker, *The Moral Basis of Property Rights* (New York University Press: New York, 1980) p. 189-190.

5. Right to consume or destroy it,
6. Right to modify it,
7. Right to transfer it,
8. Right to distribute it,
9. Right to exclude others from using it.

These rights will enable the owner of IP rights to protect his or her commercial interests in the same way as the owner of tangible property. Therefore, it is believed that this is one of the justifications for the grant of IP rights. However, it needs to balance with the other role.

3.2 IP law and its role in benefiting society

The other main accepted role of IP law is that it is used as a stimulator for the dissemination and publication of information.[20] This role has been succinctly described by the US Constitution in Article 1, Section 8, and it has been globally accepted. The purposes of the grant of IP rights are, firstly, to foster the progress of science and the useful arts, and, secondly, to encourage the creation and dissemination of information and knowledge to the public.[21] Such purposes were adopted and stated explicitly in the enactment of the US copyright Act 1909 as follows:[22]

> The enactment of copyright legislation by Congress under the terms of the Constitution is not based on any natural right that the author has in his writings, for the Supreme Court has held that such rights as he has are purely statutory rights, but on the ground that the welfare of the public will be served and progress of science and useful arts will be promoted ... Not primarily for the benefit of the author, but primarily for the benefit of the public such rights are given. Not that any particular class of citizens, however worthy, may benefit, but because the policy is believed to be for the benefit of the great body of people, in that it will stimulate writing and invention to give some bonus to authors and inventors.

---

[20] Bainbridge, supra note 17.
[21] OTA, supra note 19 at 19.
[22] OTA, supra note 19 at 3.

Thus, it can be seen that the second role of IP law is the emphasis point for the policy underlying the IP law framework. It is also the one to which the authority has to pay attention when considering extending or reducing the scope of the IP law framework. In the United Kingdom, the same policy also has long been recognised, for the justifications for copyright legislation have historically centred on the economic and social arguments.[23] It has been accepted that the protection of copyright, along with other intellectual property rights, is considered as a form of property worthy of special protection because it is seen as benefiting society as a whole and stimulating further creative activity and competition in the public interest.[24]

The second role is illustrated in the form of limitations of the legal protection provided by IP law (this may differ according to the branches of IP law) such as exceptions to the exclusive rights provided by the CDPA 1988. With relevance to this thesis, one of the exceptions is a provision in copyright law permitting the act of reverse engineering, but only to a certain extent. By allowing the act of reverse engineering, the society will benefit from the free flow of information and the dissemination of existing knowledge. This, it is believed, will encourage greater investment in the field and result in the advancement of technology.

4. THE ISSUES

The main issues in this thesis are, firstly, whether or not the current provision in copyright law that permits the act of reverse engineering is sufficient to serve the role of IP law in benefiting society. Secondly, if the current provision is not sufficient, how should the provision be amended? Thirdly, what is the impact on other branches of IP law and other related law?

---

[23] *Copinger and Skone James on Copyright* 14th ed. (Sweet & Maxwell: London, 1999) p. 30.
[24] Ibid.

5. THE THESIS

5.1 The thrust of the thesis

This thesis proposes that these two processes, namely reverse engineering and forward engineering, should be separated from each other so that the legal status of each is considered on its own merits. Nonetheless, it is submitted that the reverse engineering process should not be an infringement of copyright because it would be inconsistent with the rationale of copyright law – copyright does not protect ideas themselves but the expression of ideas. Nor should the forward engineering process be an act of infringement. This is because, in practice, whether or not this process is infringing cannot be determined until the end product has been created. For this reason, the thesis propounds that the law should focus solely on the comparison of the end products.

The thesis' proposal for the "product comparison" approach begs the question of how the courts would be able to determine whether or not infringement has occurred. This is due to the fact that the infringement issue is still a grey area so that the courts may even struggle to determine the elements in which copyright can subsist before they can reach a decision on infringement. In this respect, the doctrine of the idea/expression dichotomy is considered to be a useful tool for the courts in determining subsistence and infringement of copyright. There is uncertainty and ambiguity in the scope of this doctrine, however, as demonstrated by the fact that courts in different jurisdictions have drawn different conclusions when deciding what element constitutes idea and what is expression. An original feature of this thesis, therefore, is the suggestion as to where the focus of idea and expression should lie, and how the courts should compare the literal and non-literal elements of computer programs in relation to modern computer programming language technology such as Java and Visual Programming Language (VPL). Since it is proclaimed that Java is a breakthrough in computer programming language technology, and that Java is considered as a true object-oriented programming language, part of my thesis will examine whether or not and to what extent the existence of Java will affect the

practice of, and the need for reverse_engineering. The thesis will also consider how Java may influence the court's decision on the infringement issue.

5.2 Testing the proposed framework against the philosophy underlying IP law

My research will reveal that the framework proposed in this thesis is in line with the philosophy underlying IP law. That is to say, the proposed framework will protect the commercial interests of the owner of computer programs and benefit the public at the same time. Moreover, the basic principles of copyright law – which, as it will be seen, are undermined by the current provisions on reverse engineering – will be restored by the proposed framework.

5.3 The impact of the proposed framework on the other related areas of law

The arguments comprising the central theme of this thesis may impact upon the present non-copyright legal framework which gives protection to computer programs. The law of confidence, the purpose of which is to protect confidential information where a confidential relationship exists, is the second most important mechanism used to protect computer programs, and perhaps, the most effective way to protect ideas and confidential information. The surrender of intellectual property rights in respect of "reverse engineering" practices may raise questions about the scope and suitability of the law of confidence within such a regime. In the context of computer software, information embedded in computer programs may be considered confidential and protected by the law of confidence, so far as it is kept secret. This gives rise to such questions as whether or not the legitimate exercise of reverse engineering will result in the loss of confidentiality in the "information"; whether or not an employee can escape the obligation of confidence when computer programs are made available to the public especially on the Internet; whether or not there is a residue in the software to be protected by the law of confidence and finally whether or not the law of confidence will be capable of protecting computer programs once the copyright relaxation proposed by this thesis is established.

The proposed limitation of copyright protection may lead to attempts by leading software manufacturers to strengthen the legal protection of computer programs. Some manufacturers, who wish to maintain a 'monopoly' in their product market, may seek to divert to patent-related protection. As a result, the patent is likely to become a more attractive means of protecting computer programs because it bestows a much stronger initial protection than copyright. In the United Kingdom, although computer programs are not generally patentable *per se*, they may be patented if they are related to inventions. Therefore, the thesis will investigate how far the scope of patent protection might extend to software-related inventions and what impact this will have upon the development and exploitation of ideas.

Given its common law connection, the development of patent law in the United States is of particular interest. There have been attempts to extend the scope of US patent law to give protection to relatively broader technical effects achieved by computer programs. The thesis, therefore, will compare the United States approach with the United Kingdom approach. The thesis will also consider whether the more generous US approach towards patentability should be adopted into this country directly or indirectly by analogous reasoning. How UK patent law might develop in the aftermath of copyright regulation of reverse engineering will be assessed.

Finally, the thesis will analyse the impact of competition law on reverse engineering practices. Currently, there is a trend in the European Court of Justice towards the use of competition law to limit the exercise of intellectual property rights. It is arguable whether the operation of competition law in this respect can be effective in the sense of recognising the legality of reverse engineering without requiring changes to the provisions of the 1988 Act. Thus, the thesis will consider whether and to what extent reverse engineering can be legalised in the sense that this practice may be ringfenced by the exercise of competition law. Securing the balance between public interest concerns and the rights of the copyright holder will be considered as well as the extent to which the principles of competition law might legalise reverse engineering practices as an alternative to the relaxation of copyright.

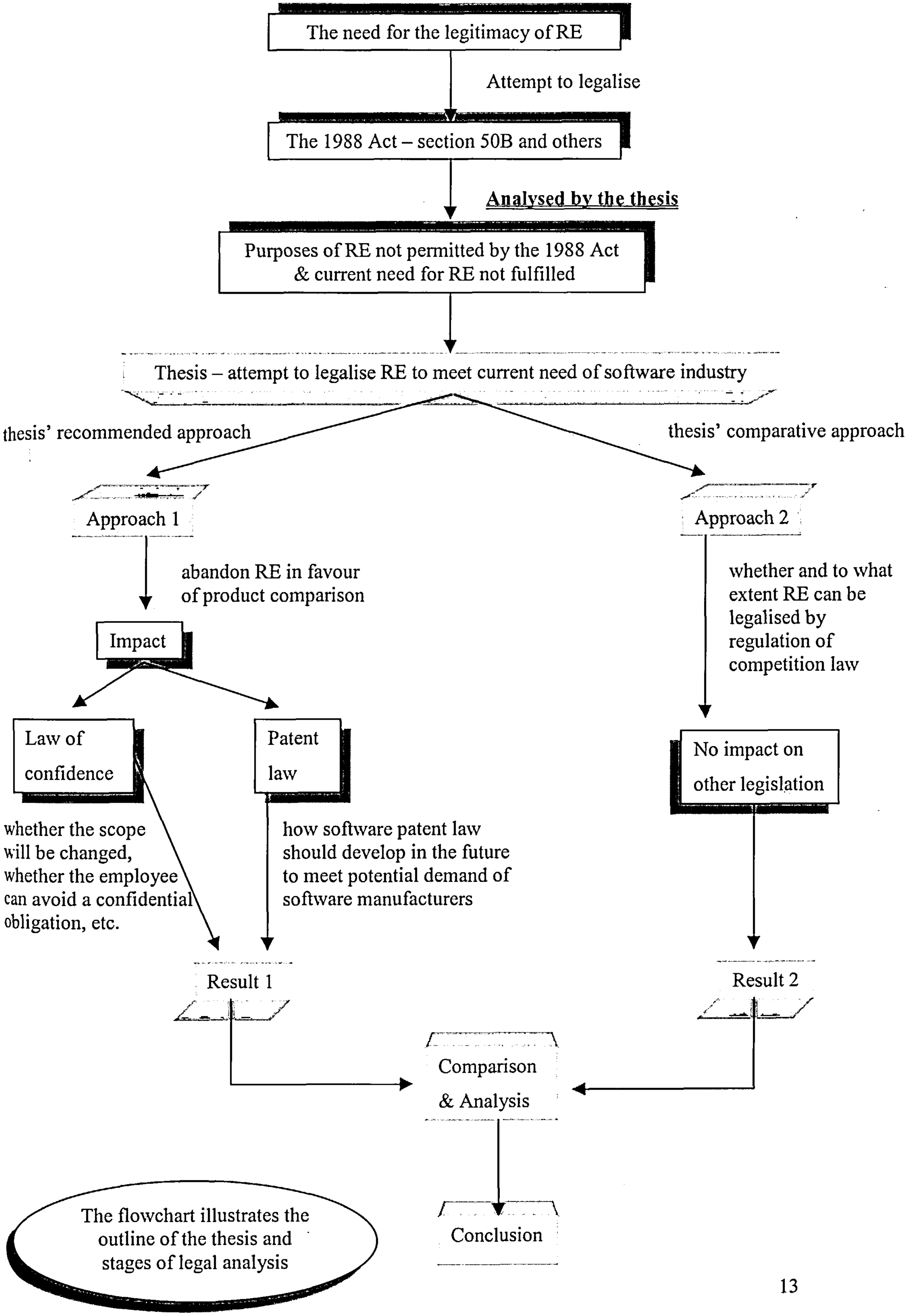
6. THE STRUCTURE OF THE THESIS

The thesis is separated into two parts. The main content of the first part begins in chapter two with the description of the nature and the current practice of reverse engineering. Legal problems presented by the practice of reverse engineering and commercial problems flowing from the exercise of copyright law are also discussed in this chapter. In chapter three, the current legal response to the solving of the legal and commercial problems will be analysed. The deficiency of the current legal response will be illustrated. In this chapter, the US response to the same issue will be discussed as it provides an alternative approach recognised by many countries. This is to consider whether or not the US approach has any deficiency and is appropriate for adoption by the EC and the UK. As will be seen, the US approach does not prove to be the most efficient solution to the problems mentioned. Therefore, this leads to the solution proposed by this thesis which is discussed in chapter four. The solution will be explained in three steps in order to show how the framework proposed by this thesis may be achieved.

In the second part, the thesis will analyse the impact of the proposed framework on other areas of law which are related to copyright law. There are three areas in this study: law of confidence, patent law and competition law. They are the subject of chapter five, six and seven respectively. In chapter five, the analysis is focused on the enforcement of the law of confidence once the proposed framework is established. The study will also include legal implications where there is and there is not an obligation of confidence. Then, in chapter six, the impact of the proposed framework on software patentability is considered. This chapter will go on to look at the forthcoming development in this field and scrutinise any implication of the proposed framework on such development. This chapter concludes by suggesting the approach towards the development of UK patent law in the future. Finally, the thesis will justify the proposed framework by showing that current competition law cannot fulfil the need for the legality of reverse engineering. This will be discussed in chapter seven.

A flowchart on the next page shows how scheme of this thesis is developed. At the end of the thesis (in chapter eight), the development of the thesis' argument from one

chapter to another will be illustrated again in a flowchart form so as to provide the best understanding of the thesis.



The flowchart illustrates the outline of the thesis and stages of legal analysis

CHAPTER TWO

PRACTICE OF REVERSE ENGINEERING AND COMMERCIAL PROBLEMS

1. INTRODUCTION

The term "reverse engineering" may be familiar to experts in the software engineering field but it may not be generally well-known to others. In layman's terms, "reverse engineering" is the process of analysing a computer program to understand how the program functions. Although the practice of reverse engineering is limited to the software industry, its implications to society are significant. However, in many legal analyses of the reverse engineering practice, the analysts fail to show the actual practice of reverse engineering. Therefore, frequently their analyses on the issue are incorrect.

This thesis, by way of introduction, begins by exploring the current practice of reverse engineering in the software industry. This is to provide a background for legal

analysis in subsequent chapters, and to demonstrate that reverse engineering is a valid practice in the software industry.

2 THE NATURE OF REVERSE ENGINEERING

The discussion in this sub-section aims to provide a factual background of computer technology regarding reverse engineering. This thesis suggests that the correct understanding of the definition and process of reverse engineering is a necessary step for the further argument and proposed solution in subsequent chapters.

2.1 Definition of reverse engineering

The term "reverse engineering", in fact, has never been defined in the Glossary of Software Engineering Terminology[1] but Chikofsky and Cross give a definition which has gained general acceptance in both public and private sectors, and is as follows:

> Reverse engineering is the process of analysing a subject system to
>
> - identify the system's components and their interrelationships and
> - create representations of the system in another form or at a higher level of abstraction.
>
> ...
>
> Reverse engineering in and of itself does *not* involve changing the subject system or creating a new system based on the reverse-engineered subject system. It is a process of *examination*, not a process of change or replication.[2]

From the definition above, it can be seen clearly that reverse engineering is simply a process of analysing an object and in this context the object analysed is a computer program. It is also clear that the sole immediate purpose of reverse engineering is to

---

[1] The Glossary of Software Engineering Terminology is provided by the Institution of Electrical and Electronic Engineers (IEEE).

[2] E J Chikofsky and Cross, 'Reverse Engineering and Design Recovery; a Taxonomy', (1990) IEEE Software 13,17. This Definition was also quoted by two leading Australian computer lawyers, Cifuentes and Fitzgerald. See Cristina Cifuentes and Anne Fitzgerald, 'Reverse Engineering of Computer programs: Comments on the Copyright Law Review Committee's Final Report on Computer Software Protection', (1995) 6 Journal of Law and Information Science 241, 248.

identify the program's components and to discern how these components interact with others. Basically, it can be said that the real purpose of reverse engineering is to understand how a given computer program works, that is to conceive the ideas underlying such a program. This is reiterated in the second paragraph of the definition quoted.

Chikofsky and Cross also include in the definition a common method of reverse engineering, namely, by creating representations of the program at higher levels of abstraction. This method is commonly known as "decompilation". It is a necessary process which has to be done in order to achieve the purpose described previously, i.e. to identify the program's components. In the context of computer programs, a representation of the program at a higher level is required in order for the programmer to understand the operation of the program clearly. This is because the representation at a higher level of abstraction produces a form of computer code which is close to human language, hence, understandable by humans.

In the software engineering field, there is another method of performing reverse engineering, which is commonly known as a "black box" method (referred to in this thesis as "black box" reverse engineering). Although this method is not described in the definition (apparently owing to its limited application) it is worth extending the discussion to its legal position. This is because it is one of the controversial issues in the "Software Directive".[3] Moreover, "black box" reverse engineering may become the only available method of reverse engineering in future because the creation of a higher level of abstraction may be technically impossible as a result of an anti-decompiler device. The process of "black box" reverse engineering will be discussed below.

2.2 The process of reverse engineering

The process of reverse engineering computer programs can be separated into two main groups according to the methods of performing reverse engineering discussed

---

[3] The Council Directive of 14 May 1991 on the Legal Protection of Computer Programs (91/250/EEC) OJ No L 122/42, 17.5.91.

above. The method of performing reverse engineering by creating representations at a higher level of abstraction dictates that the process will involve translation of computer program codes from a low level code to a higher level code. The method of making a representation of the system at a higher level is required because most computer programs distributed in the market are complied into binary code, called "object code" or "low-level code", to enable the computer to execute efficiently the instructions written in a high-level code, called "source code". The object code provides little or no information to the programmer as it represents merely electrical pulses symbolised by 0's and 1's. Hence, the underlying ideas of a computer program are hidden from inspection. If the programmer wishes to gain access to the ideas in the object code program, he would have to convert the program from object code to source code normally by using reverse engineering tools. During the process of reverse engineering, multiple copies of object code will be created by the reverse engineering tools. Source code also has to be copied several times to ensure that the final version of the source code represents the closest details to the original source code. It is also due to the fact that humans cannot remember all the million lines of code without recording it. The programmer, therefore, needs to record the code either manually or electronically. The act of converting a computer program from a low level code to a higher level code is also known as "decompilation" as mentioned above.

Another method of reverse engineering, which is called "black box" reverse engineering, involves a different process. "Black box" reverse engineering does not require a conversion of object code to source code. It can be performed by monitoring and comparing the input and output of the data entered into the computer or observing some physical result such as printing or displaying of the screen monitor. However, "black box" reverse engineering is usually insufficient to determine the underlying ideas of a complex program that is normally commercially valuable. For example, determining the underlying ideas of voice recognition software by observing inputs and outputs of voice and text data may not be sufficient to know how the

software recognises voices.[4] Therefore, the object code of the voice recognition software has to be converted into a human readable source code form.

The processes of both methods of reverse engineering are the crucial factor that renders the act of reverse engineering illegal under copyright law. In section four, the reason why these processes are considered as an infringement of copyright under current copyright law will be clarified. The section below illustrates the significance of reverse engineering in the software industry.

3. CURRENT PRACTICE OF REVERSE ENGINEERING

Reverse engineering has now been recognised as one of the major fields of computer science. Its importance has been emphasised in both the academic and practical arenas.[5] A large number of software houses and even individuals, nowadays, use reverse engineering techniques to develop a new product, to generate the profit of their product or to improve the product's quality. The practice of reverse engineering can be divided broadly into two separate areas, namely, the practice of reverse engineering in software development and in software maintenance. Each practice has its own end purposes. The differentiation of the areas of reverse engineering practices will help the legal analysis in subsequent chapters.

3.1 Practice of reverse engineering in software development

The practice of reverse engineering in software development is usually tied to the need to improve software. The need to improve software constantly arises at both the operating system and application program levels. Such a need is derived from two main groups, namely from the manufacturer of software and from the user of software. The need derived from the manufacturer is usually stimulated by

---

[4] John T. Soma, Gus Winfield and Letty Friesen, 'Software Interoperability and reverse engineering' [1994] 20 Rutgers Computer & Technology Law Journal 189, 196.

[5] In the academic arena, the emphasis on reverse engineering can be seen from "Working Conference on Reverse Engineering" sponsored by IEEE Computer Society/TCSE Committee on Reverse Engineering and Reengineering, and Reengineering Forum. The emphasis on reverse engineering in the practical arena can be seen, for example, from information regarding reverse engineering provided by the UK Government Centre for Information Systems (CCTA).

competition in the software market whereas the need derived from the user may be caused by business requirement changes.

From the manufacturer's aspect, reverse engineering is mostly used for testing the program to be released. The use of reverse engineering extends to any level of abstraction of the program and it is normally performed whenever the information is incomplete or inaccurate.[6] For instance, during the testing of a modified system, if a problem relating to interoperability occurs and the documentation does not provide the solution, reverse engineering is then performed to find out what part of the program has a malfunction. This allows the programmer to "debug" it. For example, IBM and AT&T use reverse engineering in the process of the development and improvement of their products. The practice of reverse engineering by such multinational companies can also be seen indirectly from the development of the reverse engineering tools from their laboratory. For example, the C Information abstraction System (CIA), which is a collection of reverse engineering tools for C, are developed by AT&T Labs Research.[7] This practice has paved the way for further research into reverse engineering techniques by many universities.[8]

The need derived from the manufacturer may require the programmer (who works for that manufacturer) to perform reverse engineering on software of other manufacturers in order to obtain necessary information. What can be said to be necessary information is dependent on the end purpose of each software development project. For example, if a software development project involves a creation of a program which needs to be compatible with an existing program, the necessary information retrieved by reverse engineering will be the "interface code". If the project is to create a competitive program, the necessary information will be underlying ideas and principles of how the target program functions or how screen displays are created. This can be done in the following manner.

---

[6] Andrew Johnson-Laird, 'Reverse Engineering of Software: Separating Legal Mythology From Actual Technology' [1992] Software Law Journal 331, 345.
[7] <http://www.research.att.com/sw/tools/reuse/packages/ciao.html> accessed on 16/11/98.
[8] For example, Brigham Young University <http://www.byu.edu:80/>, Virginia Commonwealth University <http://www.isy.vcu.edu/~paiken/index.htm>, University of Victoria <http://www.pigi.csc.uvic.ca/UVicRevTut/UVicRevTut.html> accessed on 11/6/99.

With regard to the first example, the existing program that is expressed in the form of object code has to be reverse engineered (converted) into source code so that the programmer can determine which parts of the existing program contain interface code. By isolating and duplicating the interface code, the programmer can develop a new compatible product by embedding this interface code in the new product. Company A, for instance, can use reverse engineering techniques to obtain the interface code and then use the interface code to create a video game that is interoperable with a video game console of company B into which various game cartridges are inserted.[9] With respect to the second example, the programmer can ascertain the ideas, principles and organisation of the program that has been reverse engineered and then incorporate these ideas into the improved program or use them to produce a new competing computer program having the same function as the original one.

It should be noted that in the circumstance where companies wish to study the products of their competitors by using reverse engineering, these companies usually perform reverse engineering secretly and may seek help from various groups of hackers from the Internet. This is because they are afraid of lawsuits since the law in this area is still uncertain.[10] At present, there are many hacker websites which provide information about reverse engineering. For example, BOMARC-Reverse Engineering,[11] The Underground Railroad,[12] Hackers Catalog,[13] The Shadow Knights[14] and 2600 Magazine.[15] There are also third-party companies which offer this kind of service, i.e. using reverse engineering to analyse the products of their customers' competitors.[16] Not only do companies reverse engineer products of others but "open source" developers also use reverse engineering to develop freely available software, such as Samba and Linux, in order to make them more reliable. Samba is a

---

[9] *Sega v Accolade* 977 F. 2d 1510 (9th Cir. 1992); *Atari v Nintendo* 975 F. 2d 832 (Fed. Cir. 1992); see Stephen B. Maebius, 'The New Use of Fair Use: Accessing Copyrighted Programs Through Reverse Engineering' (1993) 75 Journal of the Patent and Trademark Office Society 431, 432.

[10] An interview of a previous unnamed employee of Hewlett Packard.

[11] <http://w3.trib.com/~rollo/bomcat.htm>

[12] <http://www.tomah.com/samarac/>

[13] <http://www.hackerscatalog.com>

[14] <http://members.tripod.com/~The_Phantom_x/Main.html>

[15] <http//www.2600com>

[16] E-mail response, on 13/11/98, from Michael Blaha of OMT Associates, Inc. He said that he has used reverse engineering for several companies with which he consulted. However, he was not at liberty to release their names.

suit of programs which works together to allow clients to access a server's filespace and printers via the Server Message Block (SMB). This program allows Unix clients to use Microsoft file and print services.[17] In a recent survey project of John Wallberg of the Massachusetts Institute of Technology, the interviewees agreed that reverse engineering is a valid practice which is used to investigate a competitor's product and to learn about it so that the company can design and market a superior product.[18]

A high level of the use of reverse engineering for determining information or underlying ideas in a rival's program can also be seen indirectly from the development of devices used to prevent a program from being reverse engineered. Such devices are known as an "obfuscator".[19] Modern computer programs, which are written by Java programming language, are vulnerable to reverse engineering. Therefore, obfuscators are frequently contained in these programs so as to provide some measure of protection against reverse engineering.[20] A well-known obfuscator is, for example, Crema, which can obfuscate a decompiler like Mocha.[21] Nowadays, there are many decompilers and disassemblers distributed on the Internet free of charge, such as PEDasm,[22] Iasm X86,[23] and Advanced Hex Editor.[24] This indicates the popularity of the practice of reverse engineering in the software industry.

In software development the need to improve or modify software is also derived from software users (individuals and companies). Such a need comes from the fact that business requirements have changed or the user wants to increase his productivity.

---

[17] <http://sunsite.auc.dk/samba/docs/faq/sambafaq-1.html#ss1.1> accessed on 26/10/98. Samba is available at<ftp://samba.anu.edu.au/pub/samba/>. People who develop Samba can be reached at <samba-bugs@samba.anu.edu.au>. Andrew Tridgell is the person who first created Samba. He does not ask for payment, but he does appreciate it when people give him pizza in return.

[18] John Wallberg, 'Scenario from Ethics and Reverse Engineering' (last updated: 5 September 1998) <http://ethics.cwru.edu/projects/rev_scen.html> accessed on 29/10/98.

[19] D. Dyer, 'Java decompilers compared' <http://www.javaworld.com/javaworld/jw-07-1997-decommpilers.html> accessed on 25/10/98. In essence, obfuscators remove all non-essential symbolic information from the class files and, optionally, replace it with fake symbolic information designed to confuse the decompiler.

[20] Qusay H. Mahmoud, 'Java Tip 22: Protect Your Bytecodes from Reverse Engineering/Decompilation' <http://www.javaworld.com/javaworld/javatips/jw~javatip22.html> accessed on 25/10/98.

[21] Mocha is the most widely used decompiler for Java computer programs. Although the code generated by Mocha is not exactly the same as the original source code, it is close enough for someone to understand and modify.

[22] Can be found at <http://www.timeless.org.zw/software/PEDasm/PEDsm10b.zip>

[23] Can be found at <http://www.timeless.org.zw/software/Iasm/iasm.zip>

This is usually involved with bespoke software. In this scenario, in order to apply the changes correctly and efficiently, the business processes must be understood and the detailed functions of the existing software, such as the sequence of operation of modules, must be determined.[25] A problem is likely to occur if the past changes to the software cannot be well understood by the programmer. This situation normally happens when the change procedures in the past were not carried out properly. Undisciplined changes may result in poor documentation even though the existing software may give the desired end result. Poor documentation and unfamiliar code structure will eventually limit the scope of further improvement and modification. Reverse engineering, therefore, plays an important role in overcoming such limitation by helping the programmer to understand the overall structure and design of the existing software which have been rewritten over many years.[26]

Companies or organisations wishing to improve or modify their software as mentioned above may employ in-house programmers, the manufacturer of the software, or a third-party company who offers software development services. In practice, if companies or organisations employ their in-house software engineers, they may purchase reverse engineering tools available in the software market for using in the process of evaluating an existing body of code and capturing important information before making a modification. However, it is more common that a third-party company is called in to help the process of software development. This is because software development is a complicated task and requires specialist expertise. The third-party company may use its own reverse engineering tools in the process of understanding the existing code of the customers' software prior to commencing any modification or enhancement to the software. In the software industry, there are many third-party companies offering these kinds of services, e.g. Advanced Software Technologies Inc.[27] There are also many companies around the world selling reverse

---

[24] Can be found at <http://www.timeless.org.zw/software/axe/axe.zip> It is an editor for Windows 95/98/NT.

[25] CCTA by Richard West, *Reverse Engineering – An Overview* (HMSO: London, 1993) p. 5.

[26] Ibid., at p. 6. Related, but not entirely relevant, is the need for documenting the database of the legacy system. To reengineer the legacy application program, programmers may need to document the existing database first, by using a tool for reverse engineering an existing database into a physical model. The tools available in the market are, for example, ERwin/ERX from Logic Works and Embarcadero Technologies' ER/Studio, PowerDesigner from Sybase Company.

[27] <http://www.advancedsw.com/> accessed on 10/11/98.

engineering tools, for example, Applied Conversion Technologies Inc.[28] and Imagix Corp.[29] Reverse engineering tools are also being developed by individuals, e.g. Giampiero Caprino. He creates a reverse engineering compiler (REC), known as a portable decompiler.[30]

3.2 Practice of reverse engineering in software maintenance

Equally important is the application of reverse engineering to software maintenance.[31] Reverse engineering is frequently used by software engineers to check errors and to help them understand the overall structure and design of a system, particularly a large and complex software system, known as a "legacy" system. Moreover, the programmer often uses reverse engineering to analyse the quality of the code and then to evaluate the impact of future changes.[32]

Software maintenance has been recognised over the past decade as being no less significant than software development.[33] The software industry, academia and entrepreneurs have become aware that the failure of software maintenance can cause substantial financial loss and will severely affect the prosperity of their businesses. Maintaining a "legacy" system, which comprises millions of lines of codes and algorithms, and constitutes a vital asset for corporations and governments, is not an easy task. It is accepted in the software industry that software maintenance has

---

[28] This company sells a tool, called XACT. XACT is used for transforming assembly language to C. <http://www.year2000plus.com/reveng.html> accessed on 10/11/98.

[29] This company sells reverse engineering tools, called Imagix 4D. These tools are designed to use with software that is programmed by C or C++ programming language. The tools help software developers understand the legacy C and C++ system and also help them in creating design documentation, including HTML. E-mail response from Imagix Corporation on 29/10/98.

[30] The portable decompiler supports a variety of different input binary files, such as Unix Elf, Unix COFF, Windows PE, Linux and SunOS AOUT, and produces a C-like representation as output. REC – Reverse Engineering Compiler Home Page <http://www.aromatic.com/~cg/rechome.htm>.

[31] The IEEE Standard Glossary defines "software maintenance" as "the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment".

[32] West, supra note 25, at 6.

[33] The Court of Appeal in *Saphena Computing v Allied Collection Agencies* (unreported) 3 May 1989 has recognised that even when software is delivered there will still be some work to be done. The software will almost certainly contain errors and the software house will normally be expected to test the software to locate errors and make the necessary modifications. This duty will endure for a period of time though it is difficult to predict how long (*See* David Bainbridge, *Introduction to Computer Law* (3rd ed.) (Pitman Publishing: London, 1996) p. 180.

become a critical problem.[34] This is because the cost of software maintenance practices, while rarely seen as a direct cost, accounts for 50-90 percent of total life-cycle costs.[35] If the time required to comprehend software is reduced by the use of reverse engineering, it is manifest that the overall cost of software maintenance will greatly decrease.

The use of reverse engineering in the process of software maintenance has also been seen as a way to extend the life of the system and to maximise the return on the investment in the current software. This is because it can show significant saving over the cost of replacing the old system.[36] At the present time, several researchers have reported that the application of reverse engineering techniques to the software maintenance industry has yielded significant benefits to most software users.[37] Many universities and organisations have now begun to develop software reverse engineering tools to help the programmer in the process of maintaining legacy software. For instance, the computer science department of the University of British Columbia has proposed a number of reverse engineering techniques that automatically produce a high-level view of a software system.[38]

In addition to the need for the maintenance of software, the need for upgrading and migrating existing software systems has currently become more apparent and pressing too. This is because many businesses require more efficient management and higher product quality. It is suggested that reverse engineering used in software maintenance can also be done for the purpose of enhancement of the legacy system or migration to a new software platform or a new generation of hardware.[39] When it is necessary to change an old system to a completely new one, the use of reverse engineering can help in the process of changing over. This is because most new application software system developments follow on from existing systems or have to link up with

---

[34] H. Müller, 'Understanding Software Systems Using Reverse Engineering Technologies Research and Practice' <http://www.rigi.csc.uvic.ca/UVicRevTut/Abstract.html> accessed on 29/10/98.
[35] A. Frazer (P.A.V. Hall ed.), Reverse Engineering – hype, hope or here? (*Software reuse and reverse engineering in practice* (Chapman & Hall: London, 1992)), p. 221.
[36] West, supra note 25, at 23-25.
[37] J Cross, A Quilici, L Wills, P Newcomb and E Chikofsky, 'Second Working Conference on Reverse Engineering Summary Report' <http://www.cc.gatech.edu/conferences/WCRE95/summary-rpt.html> accessed on 13/11/98.
[38] <http://www.cs.ubc.ca/labs/se/projects/index.html> accessed on 10/11/98.
[39] Frazer, supra note 35, at 217.

different applications. Thus, the use of reverse engineering to check existing functionality or to define interface requirements will point to opportunities in saving time and cost.[40] In a major upgrading or migrating of a legacy system, an experienced third-party company may be called in to help in the process of changing over. For example, Ipso Facto Consulting, Inc. was asked by both UK and US large financial software houses to re-engineer their treasury systems and existing products and to position all their new developments on a client server. The aim of implementing a new environment was to cope with the treasury and market maker functions with on-line access to the dealing floors in London and New York, which processed up to 40,000 transactions per day.[41] Therefore, it can be seen that in a large re-engineering project, a company normally employs a third party rather than uses its in-house programmers.

At present, the move towards open systems and interoperability has become apparent and received a global consensus. Several multinational corporations and organisations have chosen to replace an old incompatible system with a new compatible system to incorporate tomorrow's systems and free up resources for innovation.[42] In such cases where the move towards an open system is apparent, reverse engineering may be used by a third-party company to comprehend the existing system, thus reducing the problems in changing the system to a new interoperable environment. A very good example is the role of the Open Group.[43] There are many large companies and organisations that have chosen to move towards an open system with the assistance of the Open Group. These companies and organisations are, for example, the Boeing Company, the United States Air Force, the Government of Norway (in providing its citizens with Internet-based public administration services), the UK national insurance, Litton/PRC, the US Internal Revenue Service and the UK Department of Social Security.[44]

Technically, the move towards the open system is to migrate an old legacy software system with closed environments to a new software system which can interoperate or

---

[40] West, supra note 25, at 25.
[41] This new implementation used MS-Window on front-end PCs with a Stratus back-end under UNIX running ORACLE. <http://www.ipsofacto.com/project3.htm> accessed on 10/11/98.
[42] <http://www.opengroup.org/challenge/> accessed on 10/11/98.
[43] <http://www.opengroup.org/overview/> accessed on 10/11/98.

communicate with software from other companies.[45] Seemingly, the process of transition from a legacy system to a new and compatible environment is also known as "re-engineering". The process of re-engineering normally includes two major steps, namely, "reverse engineering" and "forward engineering". Reverse engineering helps the programmer to understand the existing system before they begin the process of forward engineering to implement a new system. (Reverse and forward engineering also applies to software development in general, where a company uses reverse engineering to determine underlying ideas and uses forward engineering to create a new product. This leads to the submission that the law should recognise reverse and forward engineering and consider their legal status on their own merits.) In this circumstance, like the activities of the Open Group, it is clear that the third-party company is the party which usually uses reverse engineering in the process of moving towards the open system. In addition, when the customers wish to change a platform or an environment, the third-party companies may use reverse engineering on an application program in order to retrieve the source code and then to recompile the application program for the new platform.[46] Third-party companies may use reverse engineering not only for software migration towards the open system but also for the redocumentation of the existing system for the purpose of an efficient managing system. For example, Ascent Logic Corporation reverse engineered the Integrated Combat System of a submarine to provide a service-wide environment for managing the acquisition and maintenance of Naval systems.[47]

From the discussion above it can be clearly seen that, in practice, software maintenance does not merely involve error correction, but also includes actions taken to make subsequent maintenance more efficient and reliable, as well as the enhancement of the existing software. Therefore, it is rare that an application program is re-engineered without additional functionality being added.[48] This can be

---

[44] <http://www.opengroup.org/case-studies/> accessed on 10/11/98.
[45] John Warner, the president of the Boeing Company, expressed his concerns about the necessity of the open system, stating that 'when the people who build the equipment and write the operating systems and the applications do not comply with open standards, it makes it more difficult for us'. Moreover, he stressed that '[Boeing has] gone so far as to declare to suppliers that open systems are our direction. If that is not what you are, if that is not what your product is, then we are not interested'. <http://www.opengroup.org/comm/case-studies/boeing.htm> accessed on 10/11/98.
[46] In this circumstance, the application program is usually written in 3GL or C, C++.
[47] <http://www.year2000plus.com/reveng.html> accessed on 10/11/98.
[48] Chikofsky and Cross, supra note 2, at 16.

arguably considered as an overlap between software development and maintenance. However it might be categorised, many organisations and corporations are facing maintenance problems that mainly stem from lack of access to the source code, poor documentation and design of the existing software. Usually, the system's maintainers are not the manufacturers of the software, so the maintainers must expend many resources to examine and learn about the system by studying the source code and all documentation of the software.[49] Frequently, such source code and documentation may be lost or misplaced, or may not be available because the original software manufacture has gone out of business or just no longer supports the product. The only way to recover or retrieve source code and documents is to reverse engineer the existing software.

Reverse engineering can also be applied to facilitate safety analysis in safety-critical systems. The Bylands project,[50] for example, used program transformations developed by the University of Durham, to reverse engineer code to recover a specification in order to validate the system requirements, i.e., to ascertain that the code actually behaves as it is required to behave.[51] More recently, reverse engineering has been used by a third party to solve "the year 2000 problem" (Y2K) for which original manufacturers did not cater when creating the existing software.[52] Source Retrieval LLC., for example, offers the decompilation services for solving the Y2K compliance problem for all IBM midrange systems.[53] Indeed, it can be said that any party (e.g. manufacturers, third-party companies, users or even interested programmers) who are involved with Y2K remediation would use reverse engineering to understand the existing codes before making changes.[54]

In the software industry, there are many companies which offer reverse engineering services and which market reverse engineering tools to be used by software

---

[49] Ibid., at 14.
[50] T.M. Bull, E.J. Younger, K.H. Bennett and Z. Luo: 'BYLANDS – Reverse Engineering Safety-Critical Systems' Proc International Conference on Software Maintenance, Nice, France, 1995 (IEEE).
[51] Program Transformations, <http://www.dur.ac.uk/~dcs1ejy/Bylands/transforms.html> accessed on 26/10/98.
[52] For issues specifically relating to Y2K see Simon Halberstam and Jonathan DC Turner (eds.), *Countdown to 2000: A Guide to the Legal Issues* (Butterworths: London, 1998).
[53] Home page at <http://www.sourceretrieval.com/doccl.html> accessed on 2/11/98.
[54] E-mail response from Gerald Gannod of Computer Science Department of the Arizona State University on 12/11/98.

developers or in-house engineers. For example, Source Recovery Company offers decompilation services to IBM 360/370/380 clients who have lost their source code.[55] Software Migrations Ltd provides a service for its clients who wish to translate IBM 370 Assembler modules into equivalent readable and maintainable COBOL programs.[56] Access Research Corporation provides support and maintenance services for various network installations, e.g. Novell/MS-NT-based LANs connectivity with Internet, Electronic Name Servers (ENS), SUN/Unix, and VAX/VMS – by developing its own methodology for enhancing code maintenance, which is called Computer-Aided Reverse Engineering, to abstract information from existing code.[57]

In conclusion, it can be seen that reverse engineering is very important to the software industry and has a wide range of applications for both public and private sectors. Reverse engineering is at present a widespread practice because it is not only performed by the software manufacturer, but also by the company user and third parties or even individuals. The practice of reverse engineering is also recognised in both academic and business arenas. This indicates a potentially robust development in this field in the near future.

However, the current regulations governing the practice of reverse engineering may not be appropriate for the present situation, thereby leading to commercial problems. This will be discussed in Section five.

3.3 Impact of Java on the practice of reverse engineering

The Java programming language is regarded as a very exciting development in the field of programming languages because of its ability to deliver programs across the

---

[55] The University of Queensland, 'The General Approach to Decompilation' <http://www.csee.uq.edu.au/csm/decompilation/general.html> accessed on 26/10/98.
Clients of Source Recovery Company are, for instance, the Social Security Administration in Baltimore and Volt Information Sciences in Westbury, Australia, <http://www.accres.com/software.htm> accessed on 26/10/98. Some companies even provide for hardware reverse engineering service, such as White Rabbit Technical Services, which offers help, by means of reverse engineering, to recover technical documentation of electrical schematics, a board layout that shows component placement or circuitry, artwork for creation of PC cards, etc. Its web site is at <http://www.wabasso.com/reverse.htm>.
[56] Ibid.

Internet. The Java program can liven up the Web page, perform some sort of calculation and help the user access information stored wherever on Internet Web servers. One of the unique features of Java is that it can be used to create an application program which can run on any operating system. This is due to the fact that the Java program can be compiled to bytecodes, an intermediate form between source code and object code. With the assistant of an interpreter, a Java program in the bytecode form can be executed in any platforms.

It can be said that the advent of Java affects the practice of reverse engineering in software development in two different ways. On the one hand, it reduces the need to use reverse engineering to determine the interface code between the application program and the operating system. This is due to its platform-independent feature. On the other hand, it increases the use of reverse engineering for determining objects, underlying ideas, principles and structures of Java programs. This is because Java is a thoroughly object-oriented programming language. The unique features of the Java programming language, which directly result in the decrease and increase of reverse engineering practices, will be further explained below.

Generally, programs written in other programming languages such as Basic, C and Ada, are compiled straightaway to the appropriate object code (executable code) for a particular operating system (a Window, a Macintosh or a Unix system). A program that is compiled to the object code for the Window operating system cannot be run on the Macintosh or Unix operating system. Thus, if the programmer wishes to run the program on the Macintosh operating system, he has to recompile the program into the Macintosh-formatted object code. The programmer needs to know the format and interface code of the Macintosh operating system. If the information about the format and interface code is not readily available, reverse engineering techniques can be performed to obtain this information. Since the Java program can be compiled to bytecodes as mentioned above, it can be executed on any operating systems, provided that those operating systems have a program which can read bytecodes embedded in the systems. This program is called an "interpreter" or "Java Virtual Machine" (JVM)

---

[57] Access Research Corporation (A Division of TYX Corporation), <http://www.accres.com/software.htm> accessed on 26/10/98.

depending on what form this program resides in.[58] If this program resides in a Web browser (e.g. Netscape Navigator or Microsoft Internet Explorer), it is called an "interpreter". If it is a part of an operating system, it is known as a JVM. An interpreter enables the Web browser to download a Java program from the originating Web site and run it within the Web browser. The Web browser runs and executes the program while part of the page is being displayed.[59] It should be noted that Java programs running across the Internet are called "applets" whereas Java programs, which are not run from within a Web browser but run as "stand alone" programs,[60] are called "applications".[61] Because of its platform-independent feature, it can be clearly seen that Java decreases the need for reverse engineering in order to obtain interface code between the application program and the operating system. In other words, Java has solved the problem of the lack of interoperability between application programs and operating systems.

However, Java does not solve the problem of interoperability between two application programs. The programmer still needs to know the interface code of an existing application program if he wishes to create a new application program that is compatible with the existing application program. Therefore, the need for reverse engineering to determine the interface code of another application program still exists.

Moreover, the object-oriented feature of the Java programming language leads to the increase of the practice of reverse engineering. Because Java is a thoroughly object-

---

[58] It turns out that it is easier to write an interpreter for the Java bytecodes than it is to write a compiler for the Java source code.

[59] Hence, it is possible that a Web browser can perform as an operating system which operates applets down loaded from the Internet. This may reduce the need to buy stand alone programs. For example, we may not need to buy an MS word 97. If we want to use a word processing program, we just down load from the Internet free of charge or paying little money.

[60] This kind of programs is invoked by specifying the name of the program at the command-line interface (e.g. MS-DOS) or by clicking on a program icon in a graphical user interface (GUI).

[61] However, Java applications are currently not as popular as applications written in Visual basic (VB) or MFC programming language because the graphic toolkit supplied with Java is far too primitive to make the design task pleasant (see John Wallberg, 'Scenario from Ethics and Reverse engineering' <http://ethics.cwru.edu/projects/rev_scen.html> assessed on 29/10/98). This is also due to the fact that Java applications which are not compiled to the executable code are slower than application written in other languages (Benoît Marchal and Marc Meurrens, 'Java Decompilation and Reverse Engineering: Part I' <http://www.javacats.com/US/articles/decompiler1.html> accessed on 3/11/98). Hence Java applets are more common and should be brought into this reverse engineering discussion.

oriented programming language and capable of doing multithreading,[62] a program written in Java can be separated into many software "objects", which can be described as software building blocks. Each object is responsible for carrying out a set of related tasks. Therefore, the programmer can simply assemble objects to create a new computer program and make use of reusable components. This programming technology has resulted in a valid practice of using objects from existing programs to create a new program.[63] For this reason, the programmer often uses reverse engineering techniques to retrieve objects from existing Java programs readily available on the Internet. This is also due to the fact that Java programs in the bytecodes form can be decompiled easily, and such a practice is currently widespread. The object-oriented feature of the Java programming language is considered relevant in the context of this thesis' proposed framework because it is an important factor that will impact the courts' method of determining infringement.

To summarise, it can be seen that reverse engineering has played an important role in the software industry because it is the only way to recover information and understand how the systems work, when the source code and necessary documents as well as personnel are not available for consulting. Despite the fact that computer technology has now advanced to the point that in certain circumstances there is no need to reverse engineer operating systems in order to retrieve the interface code for creating a compatible application program, there is still the need to retrieve the interface code between application programs. Furthermore, the object-oriented programming technology leads to the increase of reverse engineering practices. Therefore, the need for reverse engineering remains apparent in the software industry.

---

[62] Multithreading is the ability for one program to do more than one thing at once, for example, printing while getting a fax. The benefits of multithreading are better interactive responsiveness and real-time behaviour.

[63] The procedure-oriented programming techniques are things of the past. The procedure-oriented techniques create what is called an algorithm. Thus, to create a program in the past, the programmer has to decide first how to manipulate the data, i.e. how to arrange the algorithm, and then decide what structure to impose on the data in order to make the manipulations easier. Object-oriented programming reverses the order. The programmer has to put data structures first, then looks at the algorithms that operate on the data.

4. LEGAL PROBLEMS PRESENTED BY THE PRACTICE OF REVERSE ENGINEERING

Leading software manufacturers argue that the practice of reverse engineering causes commercial problems to them because they may not be able to keep information contained in their computer programs secret or to hide the ideas of their programs under object code. Moreover, they are anxious that the practice of reverse engineering would lead to a high level of software piracy, in terms of piratical copying or using their creative ideas in the programs.

They also argue that their computer programs distributed in the market normally contain valuable information which their competitors target on in order to steal and use that information to produce a rival product or, in the worst case, to copy illegally and resell at very low price. Therefore, commercial problems flowing from an inability to keep ideas and information secret are that they may lose the market share owing to the increase of competing products or infringing copies of their programs. Moreover, if they lose a monopoly in the market too quickly, they would not even hope to recoup the investment that has been put in developing their products.

However, these commercial problems have been solved by copyright law, although in practice the enforcement of copyright law may not be sufficiently effective in some countries. Copyright law effectively solves these commercial problems because it provides legal protection for computer programs[64] and provides for the holder of copyright the exclusive rights that include the rights to perform acts necessary for the process of reverse engineering. Thus, copyright law makes the process of reverse engineering a breach of copyright if it is performed without an authorisation from the copyright holder, as will be explained later in this section.

In some countries, such as the United States, those commercial problems are also solved by trade secret law which provides protection for secret information against

---

[64] *Apple Computer, Inc. v Franklin Computer Corp.* 714 F.2d 1240 (3rd Cir. 1983) for the United States and the Copyright Designs and Patents Act 1988 section 3(1) for the United Kingdom.

acquisition by "improper means".[65] Therefore, the software manufacturer can use trade secret law to protect information contained in source code of their computer programs if reverse engineering is considered to be an improper means of acquiring information.[66] In principle, these two protections give a sufficient solution to the commercial problems arising from the practice of reverse engineering.

However, in the United Kingdom, the law of breach of confidence (the most equivalent of trade secret law) does not generally extend to a situation where there is no obligation of confidence. To bring an action based on the law of breach of confidence to protect information contained in a computer program, the plaintiff has to establish that there is a confidential relationship between the plaintiff and the defendant. In the case of the practice of reverse engineering by a party not in a confidential relationship, any obligation must be imposed involuntarily by the law but in the United Kingdom the law seems remarkably reluctant to intervene.[67] This is because the breach of confidence action is designed to protect the relationship of confidence between the confider and the confidee, rather than to protect the information itself.[68] Therefore, in the United Kingdom, copyright law is mainly used to prevent the practice of reverse engineering.

The exercise of these two rights by the manufacturer (the copyright holder), where applicable, may create commercial problems for small and medium size companies, the user of programs and third parties, i.e. problems arising from inability to perform reverse engineering for legitimate reasons. In the United Kingdom, this kind of commercial problem, arguably, arises chiefly from the exercise of copyright law, not from the law of breach of confidence. This is because the law of breach of confidence

---

[65] However, some controversy arises over the validity of the application of trade secret law to this area. The source of the difficulty was the explicit provision in the Constitution (Art. 1 sect. 8 ch. 8) authorising federal patent and copyright law, and its consequent enactment. In two leading cases in 1964 (*Sears Roebuck & Co. v Stifel Co.* 376 U.S. 225 (1964); *Compco Co. v Daybright Lighting Co.* 376 U.S. 234 (1964)), the Supreme Court was unanimous in holding that the supremacy of federal law implied that state law could not by-pass the restricted ambit of federal patent or copyright protection under the guise of providing remedies of a different conceptual character. *See* Colin Tapper, *Computer Law* (4th ed.) (Longman: London, 1989) p. 79-84.

[66] However, it should be noted that trade secret law in the United States is a state, not a federal issue. For example, the trade secret law in California is the California Uniform Trade Secrets Act (Dennis Campbell (ed.), *International Information Technology Law* (John Wiley & Sons: Chichester, 1997) p. 378.

[67] Chris Reed, *Computer Law* 3rd ed. (Blackstone: London, 1996) p. 230.

has a limitation in its application as stated above. Copyright law applies to most situations, even in the absence of a contract or relationship between parties. As copyright law in most countries has the same basis, this thesis chooses to base the discussion on the Copyright Designs and Patents Act 1988 in explaining why reverse engineering infringes copyright and why the exercise of copyright may create commercial problems. The thesis argues that these commercial problems destroy the balance between the public interest and the interest of the copyright holder.

4.1 Why reverse engineering infringes copyright

Reverse engineering infringes copyright because its process involves the acts restricted by copyright. In this part, two main sections of the Copyright Designs and Patents Act 1988 (the CDPA 1988) will be illustrated and considered briefly. It is helpful to make it clear that the CDPA 1988 provides for the copyright holder six exclusive rights,[69] two of which are relevant to the discussion in this part, namely the right to copy the work and the right to make an adaptation of the work. The relationship between the process of reverse engineering and these two exclusive rights will be considered below.

4.1.1 Act of reproduction

As described in Chapter one, the process of decompilation (the first method of reverse engineering) will inevitably involve reproduction of multiple copies of the object code of the target program by the decompiler, a computer program assisting the programmer in performing a reverse engineering task.[70] Hence, the exclusive right to

---

[68] Margaret Jackson, *The Development of Australian Law to Protect Undisclosed Business Information* (PhD thesis, Law School, The University of Melbourne, 1998) p. 169.

[69] Section 16: The acts restricted by copyright in a work
The owner of the copyright in a work has, in accordance with the following provisions of this Chapter, the exclusive right to do the following acts in the United Kingdom–
(a) to copy the work;
(b) to issue copies of the work to the public;
(ba) to rent or lend the work to the public;
(c) to perform, show or play the work in public;
(d) to broadcast the work or include it in a cable programme service;
(e) to make an adaptation of the work or do any of the above in relation to an adaptation.

[70] The decompiler is also referred to as a reverse engineering tool.

copy is infringed, according to section 17.[71] Although the copies created by the decompiler may not be obvious to the naked eye or may be stored temporarily in an electronic form in Random Access Memory (RAM), it is sufficient to constitute an infringement of copyright according to section 17 (2) and (6).

In respect of the process of "black box" reverse engineering, the exclusive right to copy may also be infringed, although the decompiler is not employed. This is because computer programs are normally licensed to the user and the licence is usually limited to the normal use of the program. Analysing the program by "black box" reverse engineering is arguably outside the scope of the normal use. Hence, if the program is loaded or run for the purpose of "black box" reverse engineering, it may be deemed that an unauthorised copy is created.

4.1.2 Act of adaptation

In the process of decompilation, the creation of representations of the program in another form falls squarely within the description of infringement by adaptation in section 21.[72] The act of converting a computer code either from object code to assembly code or from assembly code to source code falls within the meaning of "translation" which is in turn included in the definition of "adaptation".[73] The process of "black box" reverse engineering, on the other hand, does not involve the conversion of object code at all. Thus, it will not infringe the exclusive right to make an adaptation of the work.

---

[71] Section 17: Infringement of copyright by copying
(1) The copying of the work is an act restricted by the copyright in every description of copyright work; and references in this Part to copying and copies shall be construed as follows.
(2) Copying in relation to a literary, dramatic, musical or artistic work means reproducing the work in any material form.
This includes storing the work in any medium by electronic means. …
(6) Copying in relation to any description of work includes the making of copies which are transient or are incidental to some other use of the work.

[72] Section 21: Infringement by making adaptation or act done in relation to adaptation
(1) The making of an adaptation of the work is an act restricted by the copyright in a literary, dramatic or musical work. …
(3) In this Part "adaptation"– …
(ab) in relation to a computer program, means an arrangement or altered version of the program or a translation of it …
(4) In relation to a computer program a "translation" includes a version of the program in which it is converted into or out of a computer language or code or into a different computer language or code.

4.2 Legal problems

The very legal problem of protecting a computer program under copyright law is that the classification of a computer program as a literary work results in the extension of the scope of copyright protection to ideas behind the computer program. This is because the practical means of accessing the ideas of a computer program is via reverse engineering which is technically prohibited as it infringes the exclusive right of the copyright holder as shown in section 4.1. As it is recognised internationally that copyright protects expression, not the ideas of expression,[74] exercising copyright to forbid the practice of reverse engineering goes far beyond the scope and intention of copyright law.

The exercise of copyright to prevent an access to the idea in a computer program also creates inconsistency in copyright protection of literary work. While traditional literary works like books of fiction allow the user (the reader) to access the ideas easily, the modern literary work which is a computer program does not allow so. Therefore, it can be seen from the comparison between traditional literary works and computer programs that the extension of copyright protection to computer programs distorts the main principle of copyright law, namely that only expression of the idea is protected, not the idea itself.

5. COMMERCIAL PROBLEMS FLOWING FROM THE EXERCISE OF COPYRIGHT LAW

According to the legal problems of reverse engineering discussed above, the following questions are whether the legal problems will lead to commercial problems in the software industry and, if so, who are affected by the commercial problems, and whether this will affect the economics of the community. In short, the issue is

---

[73] See (general discussion and comparison to previous Acts) Henry Carr and Richard Arnold, *Computer Software 2nd ed.* (Sweet & Maxwell: Oxford, 1992) p. 91-103.

[74] See M. Lehmann and T. Dreier, 'The Legal Protection of Computer Programs: Certain aspects of the Proposal for an (EC) Council Directive' [1990] Computer Law & Practice 92, 94. They argue that the distinction between the freedom of ideas and the exclusive protectability of their expression is well known to all continental copyright systems, such as France, Italy and Germany.

whether the inadvertent protection of a program's ideas will unfairly hinder the advancement of innovation in the software industry.[75]

This thesis endeavours to illustrate that the answer will be in the affirmative because it can be seen that the pendulum of protection has swung from being inadequate to a level that impedes competition.[76] To answer the issues clearly, the thesis begins with the discussion of commercial problems stemming from the protection of ideas (by not permitting reverse engineering). This will be followed by the discussion as to whether the commercial problems need to be solved. The economic reason will support a cogent argument that these commercial problems need to be solved as quickly as possible because inappropriate legal protection for computer programs can affect the pace of technological advancement in the software industry and the extent to which this advancement is disseminated and used in the economy.[77]

The commercial problems flowing from the undue exercise of copyright can be separated into two areas, namely, commercial problems in software development and in software maintenance. Since the software manufacturer can now protect its software under copyright law, the parties who suffer from commercial problems can be said to be the user of software and the third party who wishes to enter the same market. Thus, this thesis defines "commercial problems" as the inability of individuals and companies to perform software reverse engineering legally for their businesses, which may lead to financial difficulties.

5.1 Commercial problems in software development

To analyse commercial problems in software development efficiently and realistically, this thesis looks at the software development market as consisting of three groups. The first group consists of leading companies or major software

---

[75] Gary R. Ignatin, 'Let the Hackers Hack: Allowing the Reverse Engineering of Copyrighted Computer Programs to Achieve Compatibility' (1992) 140 University of Pennsylvania Law Review 1999, 2015.

[76] The CLRC Recommendations on Reverse Engineering and Decompilation: Giving Local Developers an Equal Right to Compete (Executive Summary) <http://www.sisa.org.au/SISASubmission1.html> accessed on 25/5/98. See also the Powerflex case.

[77] 'The OTA Report – Background Paper – Summary, Overview and Issues' [1990-91] 4 The Computer Law and Security Report 11.

manufacturers who dominate the software market. For the ease of understanding, the first group is referred to as the "insider". The second group comprise small and medium size companies which have a minor share in the software market and which attempt to enter the market. They are referred to as the "outsider". The last group is called the "consumer" who may be companies or individuals who use software of the insider for their businesses.

In this analysis, the thesis assumes that the insider is free of commercial problems arising from the practice of reverse engineering because it can employ copyright law to prevent such practice by others. The thesis now turns to consider what are the commercial problems for the outsider and the consumer in not being able to perform reverse engineering.

For the outsider, the prohibition of reverse engineering will lead to three main direct commercial problems. There are as follows:

i. inability to develop interoperable products;
ii. inability to develop competing products; and
iii. inability to develop other products which are based on the same ideas and principles but which do not compete or are not interoperable with the product of the insider.

These three problems are derived from the same basis, that is, the outsider cannot ascertain ideas, principles and interface specifications of the product of the insider by using reverse engineering techniques. Without the assistance of the reverse engineering technique, it would be impossible for competitors to develop competing software products since it is necessary to understand how a program functions before one can develop a competitive or compatible program,[78] and frequently external constraints often dictate conformity with industry's standards of format and expression.[79] These commercial problems will prevent the outsider from entering the

---

[78] Kathleen Gilbert-Macmillan, 'Intellectual Property Law for Reverse Engineering Computer Programs in the European Community' (1993) 9 Computer & High Technology Law Journal 247, 256.
[79] Linda G. Morrison, 'The EC Directive on the Legal Protection of Computer Programs: Does It Leave Room for Reverse Engineering Beyond the Need for Interoperability' (1992) 25 Vanderbilt Journal of Transnational Law 293 at 324. Interoperability can only be achieved if the developer knows the rules

market or from retaining its small market share. This is because the consumer may be locked into the product of the insider and the only way for the outsider to enter the market is to create a program which conforms to the *de facto* standard created by the insider's product, that is to produce a product compatible with the insider's product.[80] A number of the European Community-based outsider operates in the software market by developing add-on or substitute software products designed to interoperate with the products of the American outsiders or to emulate their functionality.[81] These outsiders inevitably rely upon a degree of openness and the ability to analyse competitor's products by reverse engineering, which leads to copying of portions of code and in some instances whole programs.[82] The undue exercise of exclusive rights by the insider may result in the hindrance of the development of a new product by the outsider. This is because in the software industry the nature of developing a new product usually requires building upon the ideas and functions of existing product,[83] which is in turn assisted by the practice of reverse engineering to glean those ideas.

The outsider may also face difficulty in attempting to retain its market share because the insider may slightly change the standard so that the existing product of the outsider is not compatible with the insider's product which still holds a majority of the market share. Moreover, the commercial problem, which arises from the inability to make use of existing resources such as ideas and principles of the product of the insider, may result in unnecessarily greater capital being required for developing a new program based on existing ideas and principles. Frequently, software licensing for the use of specifications necessary for creating a new compatible product may incur unreasonable expense to the outsider. Therefore, inability to perform reverse engineering lawfully may prevent small companies from developing programs of the same type (which may effectively prevent them from entering the market) or creating a new section of the software market.

---

of interconnection with the other systems (The CLRC Executive Summary, <http://www.sisa.org.au/SiSASubmission1.html>).

[80] Gilbert-Macmillan, supra note 78, at 250.

[81] Eric Alexander Dumbill, 'EC Directive on Computer Software Protection' [1991] Computer Law & Practice 210.

[82] Jonathan Owens, 'Software Reverse Engineering and Clean-Rooming, When Is It Infringement?' (1993) 9 Computer & High Technology Law Journal 527, 528.

[83] John A. William, 'Can Reverse Engineering of Software Ever Be Fair Use: Application of *Campbell's "Transformative Use"* Concept' (1996) 71 Washington Law Review 255, 269.

Finally, these commercial problems may lead to a decrease of competition, which, in the end, may create a monopoly in the software market. A monopoly in the market may result in few varieties of products and retard the development of technology.[84] It is suggested that when a monopoly occurs the incentive of the software manufacturer to develop a better product will be reduced.[85] Hence, the advancement of technology will be stifled. Thus, it can be concluded that over-enhanced protection is likely to impair competition and innovation, in so far as it limits the opportunities to write software which can interoperate with, or emulate, the functions of the products of the insider. This situation is not only in conflict with competition policy but also in contrast to the intention of copyright law which aims to promote the development of science and technology.[86]

From the consumer's point of view (companies and individuals), the exercise of copyright law to prevent the practice of reverse engineering will create the following problems:

i. inability to develop (enhance) the software purchased (licensed) from the insider; (this category overlaps with the first category of software maintenance)
ii. inability to develop their own software that is compatible with the purchased software;
iii. inability to create their own software based on the purchased software for their own use.

All three types of inability can be described as commercial problems because they may result in the consumer's inability to improve productivity, to improve efficiency in management, or to improve the quality of the product.

---

[84] Lawrence D. Graham and Richard O. Zerbe, Jr., 'Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure' (1996) 22 Rutgers Computer & Technology Law Journal 61, 71-78.
[85] Dumbill, supra note 81, at 210.
[86] '[C]opyright law does not give rise to monopolies. At least, in principle, it should not do so.' See David Bainbridge, *Software Copyright Law 3rd ed.* (Butterworths: London, 1997) p. 13. See also U.S. Constitution Article I, § 8, cl. 8.

In many instances, the consumer's need to improve productivity of his goods is frustrated because his software system cannot be enhanced as his programmers or a third party company cannot understand the system well enough to modify the system. From time to time, the consumer may need to produce an additional program which can interoperate with the system that he purchases from the insider in order to improve productivity or to improve efficiency in management. Moreover, the enhancement of the purchased system or the creation of a new program, compatible with the purchased system, may be done for the purpose of improving the quality of the product itself. This is to maximise the return on the investment in the purchased system. Maximising the return can also be done by developing, for the consumer's own use, his own program based on the purchased system. The program developed as such can be designed to be used for the same purpose or a different purpose which may not affect the business of the insider.

The three types of the consumer inability may create a chain of commercial problems that affect the business of a third party company who offers software development services. Normally, the third party company is called in to help in the development of the system, particularly the legacy system. The prohibition on the practice of reverse engineering may considerably damage the business of the third party company as there may be no other way to understand the system before making changes and development. The consumer is also normally limited by a software contract to perform the acts of copying and adaptation so far as it necessitates the use of the program only, but not to authorise a third party company to do the acts restricted by copyright. In addition, the software licensing usually states clearly that reverse engineering is not permitted.[87] Therefore, the consumer and a third party company are not only restricted by copyright law, but also bound by contractual obligations not to perform reverse engineering.

To sum up, it can be seen that inability to perform reverse engineering legally will lead to many different commercial problems and the problems are not limited to the software industry. It extends to other businesses which rely heavily on computer systems.

5.2 Commercial problems in software maintenance

Commercial problems in software maintenance may also arise from the exercise of exclusive rights by the manufacturer who supplies the software. In the process of software maintenance, the understanding of the system is crucial. Without the use of reverse engineering techniques, software maintenance tasks could not be executed easily, if the documentation of the system is not complete.[88]

Frequently, the software maintenance task is not carried out by the manufacturer who supplies software, but by the end-user company's programmers or a third party company who specialise in maintaining software systems. If the end user or the third party company cannot perform reverse engineering, it means that they may not be able to maintain their system properly, particularly if the system is a "legacy" system.[89] For the purpose of analysis in this part, the "insider" means the software manufacturer who supplies software to the consumer and provides a maintenance service as part of its business. Companies being the "outsider" are those which offer maintenance services in competition with the insider.

The main commercial problem flowing from a ban on reverse engineering is that the consumer may not be able to perform software maintenance for his own software and the outsider may not be able to carry out software maintenance services.[90] However, the term "software maintenance" is too wide to define specific commercial problems. Therefore, the term "software maintenance" needs to be clarified. The IEEE Standard Glossary defines software maintenance as:

> The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment.

---

[87] However, because of the legal response to these problems, many software contracts have been amended accordingly.
[88] Gilbert-Macmillan, supra note 78, at 259.
[89] Please see the definition of "legacy system" in the abbreviations & definitions part at the beginning of this thesis.
[90] Clifford Chance, *The European Software Directive* at p. 7.

According to this definition, software maintenance can be classified into four types – perfective, corrective, adaptive and preventive. Each of these types of software maintenance requires the understanding of the detailed aspects of the system prior to making changes to the system. Therefore, the ban on reverse engineering may result in four different commercial problems as follows:

(The consumer and the outsider are)

i. ***perfective:*** unable to modify or enhance the existing functionality or performance of software; (this type of problem overlaps with the problem in software development as discussed previously)
ii. ***corrective:*** unable to correct error;
iii. ***adaptive:*** unable to make changes to software necessary to adapt it for a change of the supporting environment, network or hardware platform;
iv. ***preventive:*** unable to take actions necessary to make subsequent maintenance of application software more efficient and reliable.

These commercial problems may result in, first of all, the loss of investment in the system. In some cases, the consumer may not yet have recouped any benefit from his investment in the current system. When the system breaks down, he will need a large budget to restore the system or to purchase a new one. This makes the consumer not only unable to maximise the return on the previous investment, but also having to spend capital for a new investment, which may affect its financial stability.

Moreover, if the system breaks down due to poor maintenance, the consumer may lose data contained in the defective system and this will incur expense in re-entering the data. If the consumer is involved in servicing businesses, the failure of maintenance of software may affect his management efficiency, thereby, the standard of his company may not be maintained. At present, competition in most market sectors is very high and the success of many companies relies on the efficiency of its computer system. Therefore, commercial problems derived from an inability to maintain software cannot be ignored since such commercial problems may put companies at the risk of being liquidated.

Commercial problems, which arise from the inability to change or modify software for a new environment, network or platform, may cause the consumer to encounter difficulties by being rejected from other companies which employ a different technological standard. For example, if the communication among companies or between the consumer and his clients needs to be through an open system and the consumer is unable to adapt his system to conform to such a system, he will have difficulty in operating his business. He may then be forced to go out of the business.

Finally, for the "preventive" maintenance, the consumer's inability to analyse the quality of the code whilst maintaining the system in order to evaluate the impact of future changes or to make subsequent maintenance of the software more efficient and reliable – may have a far-reaching effect. An application of "preventive" maintenance is obvious in the late 1990s. Y2K problem is a good example for the need for "preventive" maintenance. If the consumer could not prevent this potential error, the problem would clearly affect their software system which in turn would affect the management of the company. Therefore, it can be seen that if the practice of reverse engineering is not permitted, many companies will not be able to prepare themselves for the impact of the future changes and this will affect the prosperity of the companies.

6. CONCLUSION

From the commercial problems described above, it is necessary to consider whether those commercial problems need to be solved. The factor, which is frequently used to evaluate the appropriateness of legal intervention to solve the commercial problems, is usually the economic impact of the commercial problems.[91] If the commercial problems do not badly impact upon the economics of the society as a whole, then there is no need of legal intervention. On the other hand, if they actually affect society in the way that the public interest is prejudiced, such legal intervention becomes necessary. It is suggested that the public interest be best served by encouraging technical progress, even where it means that original works can be

[91] It is sometimes said that enormous economic importance is attached to decompilation or 'reverse engineering' of interfaces. See Andreas Wiebe, 'European Copyright Protection of Software from a German Perspective' (1993) 9 Computer Law & Practice 79, 83.

reproduced, both cheaply and easily. It is essential to foster a legal environment which favours creativity, innovation and competition.[92] Therefore, in this context, the underlying rationale is that there should be no overprotection of intellectual industrial quasi-monopoly-rights. This would avoid the possibility of there being a barrier of entry for competition and, more generally, would never hinder competition more than is absolutely necessary to allow the author his adequate market participation.[93]

As can be seen from the discussion in the previous section, the commercial problems (inability to enter market in both software development and software maintenance) will directly result in a decrease of competition in the software market. The European Committee for Interoperable Systems (ECIS) confirms that the strong protection of computer programs that creates the commercial problems would have the effect of stifling innovative competition from, at least, interoperable products.[94] ECIS illustrates clearly from the business perspective that the small and medium size companies' commercial problems will result in a reduction of competitive supply of compatible products. This will also inhibit the growth of small and medium sized European suppliers of computer products intended for open systems, and jeopardise the future entrepreneurial producers of computer products who depend upon access to interfaces in developing innovative and competitive products.[95] Likewise the EC Commission appears to be aware of the fact that a strong copyright protection would reduce competition in the software industry.[96] The decrease of competition would eventually create a *de facto* monopoly in the software market; an unwanted situation that the EC seeks to avoid. Some monopolies, it can be said, are extended to services beyond the scope of copyright, such as maintenance or systems integration.[97]

Monopolies in the software market are considered undesirable because they will hinder free exchange of ideas and, therefore, stifle innovation.[98] Moreover, because a monopoly will curb competition on proprietary systems, it will adversely affect the

---

[92] European Commission Green Paper on Copyright [(1) COM (88) 172], [1988] 3 The Computer Law and Security Report 8.
[93] Lehmann and Dreier, supra note 74.
[94] Mark Powell, 'The Software Directive' p. 5.
[95] ECIS, 'The Proposed EC Directive on the Legal Protection for Computer Programs: Position Statement' [1990] Computer Law & Practice 97, 98.
[96] Lehmann and Dreier, supra note 74, at 93.
[97] Powell, supra note 94, at 5.
[98] Gilbert-Macmillan, supra note 78, at 260.

emergence and growth of open systems, and the creation of an open market for the information industry. This will in particular deprive European consumers of access to innovative and competitive products because the European software industry will be placed at an extreme disadvantage in the international marketplace.[99] It will also thwart the maintenance of open interfaces in other industries, such as in the telecommunications industry.[100] As computer programs have been essential to recent advances in biotechnology, communications, transportation, manufacturing and virtually every other field of study, the improper protection of computer programs will have ramifications far beyond the perimeters of the software industry.[101] Therefore, it can be seen that the legal protection of computer programs without permitting reverse engineering will cause the copyright holder's right and the public interest to become unbalanced because copyright protection can be used by the copyright holder as a mechanism to impede competition and extract monopolistic profits from consumers.

In solving such problems, the underlying policy is normally consulted and it has been widely accepted that the law should permit reverse engineering. This is because although the software manufacturer may lose the exclusive right to control the derivative market and the maintenance market, society would gain from the increase of competition which would advance the technology.[102] If society's gain outweighs the loss to the copyright holder, the justification of reverse engineering is deemed to be satisfied.[103] Indeed, authorities in many countries have agreed that permitting reverse engineering will encourage the free flow of non-protected ideas and stimulate the creation of new forms of expression of those ideas for the public benefit[104], and

---

[99] ECIS Position Statement, supra note 95, at 98.
[100] Ibid.
[101] Ignatin, supra note 75, at 2022.
[102] Paul Durdik, 'Reverse Engineering As a Fair Use Defense to Software Copyright Infringement' (1994) 34 Jurimetrics Journal 451, 465.
[103] Morrison, supra note 79, at 323. She refers to J.H. Reichman, 'Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research' (1989) 42 Vand. L. Rev. 639, 699 n. 312
[104] The CLRC Executive Summary p. 7 of 11. It is also generally accepted in US copyright law that compatibility within the computer industry is in the interests of the general public (Daniel Hayes, 'London Branch Report: Reverse Engineering – The American Experience' [1994] Computer and Law 13).

will be consistent with the purpose of copyright, i.e. to promote the progress of science and the useful arts.[105]

However, the extent to which reverse engineering is permitted varies from one country to another. This thesis will examine in the next chapter whether the extent to which reverse engineering is permitted in the European Community, the United Kingdom and the United States, can solve the commercial problems addressed above.

---

[105] U.S. Const. Art. I, § 8, cl. 8. It can be seen that the potential rewards of open systems, which are the result of the legality of reverse engineering, are so great that many organisations within central and local governments and the armed forces have already started to put in place open systems strategies. Many private sector organisations are now moving in the same direction (Clifford Chance, *The European Software Directive*, p. 8).

CHAPTER THREE

THE CURRENT LEGAL RESPONSE AND ITS DEFICIENCY

1. INTRODUCTION

From the previous chapter, it can be seen that the commercial problems described would severely affect the economics of society and stifle the advancement of computer technology. This chapter will analyse the current legal response to those commercial problems and consider whether the legal response is sufficiently adequate to keep the economics of society on its toes.

For the ease of analysis, the thesis classifies the commercial problems as follows

1. Inability to develop interoperable products
2. Inability to develop competing products
3. Inability to develop other products
4. Inability to perform perfective maintenance (together with the enhancement of software)

5. Inability to perform corrective maintenance
6. Inability to perform adaptive maintenance
7. Inability to perform preventive maintenance

The commercial problems number 1-3 are derived from software development and the problems number 4-7 derived from software maintenance. In this chapter, three jurisdictions will be subject to analysis. The Software Directive is analysed because it is the legislation on which the UK Copyright Designs and Patents Act 1988 is based. US copyright law is analysed because it provides a different approach which is considered by many countries as providing the most flexibility to cope with the development of the computer technology and which was once considered by an authority in the UK as a suitable approach for the United Kingdom.

2. SOFTWARE DIRECTIVE

On 14 May 1991, the European Council formally adopted the Council Directive on the Legal Protection of Computer Programs[1] (referred to in this thesis as the "Software Directive") which provides regulation governing, and which arguably legalises, reverse engineering practices. The Software Directive gives direct answers only to two problems, namely, the problem concerning inability to develop interoperable products (problem no.1) and the problem concerning inability to perform corrective maintenance (problem no.5). The Software Directive does not give any direct answer to the rest of the problems. Therefore, this thesis will analyse the provisions of the Software Directive and consider what may be the answer, albeit indirectly, for the rest of the problems. In analysing the provisions, the thesis will consider whether the legal reasoning underpinning the answers is appropriate or not, and will consider the practical results flowing from those answers. Accordingly, the analysis in this part will be separated into two groups. The first group involves an analysis of the direct answers given to the two problems (no.1 and 5). The second group involves an analysis of the unsolved problems to which the Software Directive may give indirect answers.

---

[1] The Council Directive of 14 May 1991 on the Legal Protection of Computer Programs (91/250/EEC) OJ No L 122/42, 17.5.91.

2.1 The direct answers

As stated above, there are only two problems directly answered by the Software Directive, namely, the problem concerning inability to develop interoperable products and the problem concerning inability to perform corrective maintenance.

2.1.1 Inability to develop interoperable products

The Software Directive gives an answer in Article 6, which permits decompilation performed for the purpose of interoperability. The legal reasoning underpinning Article 6 is based mainly on economic reasons derived from the opinion given by interested parties in the software industry. Article 6 originated from the fact that initially the proposed draft of the Software Directive was silent on the reverse engineering issue. It was suggested that this would amount to a ban on the practice and it is also suggested that the Software Directive should contain an express provision permitting reverse engineering.[2] This led to an intensive debate (as to whether reverse engineering should be permitted) between two powerful groups – the Software Action Group for Europe (SAGE)[3] and the European Committee for Interoperable Systems (ECIS)[4] – representing diverse interests in the software industry. The former was in favour of the strict protection of computer programs but the latter strongly suggested that reverse engineering should be permitted to enable small and medium size companies to enter the software market.

SAGE argued that legalising reverse engineering would cause a significant diversion of existing law. Moreover, reverse engineering was in fact unnecessary for developing interoperable products because information relating to the parts necessary for interoperation was available through published materials, like manuals, or on request. Finally, and perhaps most importantly, SAGE contended that, unlike

---

[2] ECIS, 'The proposed EC Directive on the legal protection for computer programs: Position Statement' [1990] Computer Law & Practice 97, 99.
[3] SAGE comprises mainly of the United States firms such as IBM, DEC, Microsoft, WordPerfect and Lotus.
[4] ECIS represents small to medium sized software and hardware manufacturers e.g. Frances' Group Bull, ICL, Fujitsu and also IBM's competitors such as Unisys Corp and Sun Microsystems.

hardware, software could be translated and duplicated easily.[5] By legalising the reverse engineering practice, "clone" programs could be reproduced at much lower costs than the original. Thus, the lead time normally enjoyed by the owner of copyright in the original program would be so dramatically reduced that he would no longer be able to recoup the research expenditure during the period of temporary monopoly provided by the lead time.[6] This appeared to be the factor that made SAGE believe that the legality of reverse engineering would discourage software development and innovation.[7]

On the other hand, ECIS argued that a provision allowing reverse engineering activities was needed to enable competitors to determine the ideas underlying successful computer programs of the established vendors. It suggested that without such a clear provision it would be extremely difficult, if not impossible, for competitors to develop products or systems that were compatible with existing programs, since it was necessary to understand how an existing program functioned before a new interoperable program could be developed.[8] In other words, it was necessary to obtain, or perhaps copy, the "interface" code of the existing program in order to create an interoperable product.

From the economic aspect, ECIS argued that because established software markets were dominated by industry giants like IBM, DEC and Microsoft, only by producing computer programs which were interoperable with the market leaders' products, could software newcomers or small and medium sized companies even hope to compete.[9] It was not feasible for these small companies to enter the market by introducing a new

---

[5] Kathleen Gilbert-Macmillan, 'Intellectual Property Law for Reverse Engineering Computer Programs in the European Community' (1993) 9 Computer & High Technology Law Journal 247, 257.
[6] Linda G. Morrison, 'The EC Directive on the Legal Protection of Computer Programs: Does It Leave Room for Reverse Engineering Beyond the Need for Interoperability' (1992) 25 Vanderbilt Journal of Transnational Law 293, 303. See also Anthony L Clapes, 'Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs, (1987) 34 UCLA Law Review 1493, 1509. For example, the authors write:
'Software is different. There are typically no manufacturing processes to analyse, and no special factories to set up. Software is written and tested; it is then published, like books, records, or videotapes. It is possible to copy a computer program in seconds and readily reproduce that copy by the hundreds of thousands. It is more difficult, but nonetheless relatively easy, to adapt, translate, or "port" a program, and thereby appropriate much of the value inherent in the original author's creation. Software, by its nature, lends itself to quick and unexpected duplication and even translation.'
[7] Morrison, Ibid., at 303.
[8] Gilbert-Macmillan, supra note 5, at 256.
[9] Morrison, supra note 6, at 301.

breed of system which was not based on the existing *de facto* standard, since this would involve a large amount of investment and involve an impractical task of persuading customers to be familiar with a new environment and to re-enter all of their data and files into the entirely new system. Furthermore, ECIS was convinced that the ban on reverse engineering activities, carried out for legitimate purposes as opposed to piracy,[10] would retard the evolution of the software industry. The view of ECIS was supported by Computer Users in Europe (CUE),[11] another lobbying group in favour of less restrictive copyright protection, who pointed out that silence on the legality of reverse engineering would hinder the development towards the "open systems".[12] CUE noted that the lobbying efforts of SAGE showed an attempt by dominant US computer companies to convert their leading commercial positions into legal monopolies.[13] Therefore, ECIS concluded that reverse engineering should be legalised to allow competitors carrying out reverse engineering on a successful product to determine file structure and format in an effort to promote compatibility and the "open system".[14]

The Commission was persuaded by the ECIS' rational argument because the argument was consistent with the aim of the Commission which originally indicated that the goal of the Software Directive is to enable small and medium sized enterprises to keep pace with other industrialised states.[15] In the Memorandum, the Commission saw the promotion of interoperability and the development of open systems strategies as keys to promoting competition in the software industry.[16] The

---

[10] The most widespread form of software piracy is wholesale copying. For example, Lotus claims that over half of its potential sales of 1-2-3 are lost to pirates, at a cost of millions of dollars every year. The US-based Business Software Alliance (BSA) claims that while 40 percent of US software packages sold in the United States are illegal copies, this figure rises to over 75 percent in Japan, Germany, Belgium, Holland, and Italy and to 80-90 percent in China, Taiwan, Korea, Spain, Portugal and Switzerland. *See* Tom Forester and Perry Morrison, *Computer Ethics 2nd ed.* (The MIT Press: Cambridge, 1994) p. 52-57.

[11] CUE includes large computer users, as opposed to software producers, such as Barclays Bank, Galileo International, the European airline reservation system, and the West German Aerospace Research Centre.

[12] Open systems "allow machines and software from different manufacturers to be used together". CUE drew an analogy between compatible open systems and a hi-fi system: when buying a stereo system, one can buy components from various companies. CUE argued that the same should be true for computer systems.

[13] Morrison, supra note 6, at 303.

[14] Ibid.

[15] Commission Proposal for a Council Directive on the Legal Protection of Computer Programs, COM (88) 816 final, 1989 O.J. (C91) 4, 5.

[16] See also, Recital 9.

argument of ECIS prompted the Commission to realise that reverse engineering was necessary for creating an interoperable product.

This thesis agrees that the legal reasoning based on the ECIS' economic argument is convincing. A problem is likely to occur if SAGE's protectionist approach were to be followed. The problem is that a successful software developer, whose products become an industry standard, may create a *de facto* monopoly. The possibility of gaining a stronghold on the software market prompts vendors, such as IBM and Lotus, to compete fiercely for the initial sale. After this first sale, the user becomes locked into products by this vendor unless compatible products are produced.[17] For example, in the context of a computer spreadsheet, a user who has invested time and effort to learn a particular spreadsheet interface faces an inertia against switching to another vendor's spreadsheet program with different interfaces.[18]

Furthermore, the user may have accumulated spreadsheets containing masses of accounting and other critical data using the data structures and formats specified by the software developer. The cost of re-entering or recompiling this data into a different format makes switching spreadsheet vendors economically unfeasible, unless some standardisation of interfaces is permitted by copyright law.[19] Once this dependence on a single product manufacturer has been created, the dominant vendor can set price levels greater than cost without losing customers.[20]

Even if compatibility becomes a stated policy goal, the leading vendor may still gain significant competitive advantage by slightly changing the standard with little or no notice.[21] These slight changes suddenly make competitors' products no longer compatible; the industry second comer again finds itself struggling to maintain any market share that it had gained through the development of compatible components and software. Therefore, it is obvious that legalising reverse engineering practice will

---

[17] Joseph Farrell, 'Standardization and Intellectual Property', (1989) 30 Jurimetrics Journal 35, 38-39.
[18] Thomas M Hemnes, 'Three Common Fallacies in the User Interface Copyright Debate, 6 Computer Law & Practice 163, 167.
[19] Michel Colombe & Caroline Meyer, 'Seeking Interoperability: An Industry Response', (1990) 12 European Intellectual Property Review 79, 80.
[20] Farrell, supra note 16, at 38.
[21] Ibid., at 40.

promote competition and will keep the entire software industry on its toes.[22] Under this assumption, the only party that may suffer economic loss is the software producer holding the copyright, which often will be a large software manufacturer.[23] Therefore, this approach will leave ultimate benefit to the public because the software industry would evolve and improve without inflating prices.

However, because one of the goals of the Commission was also to protect software from unauthorised copying[24] and SAGE had shown a cogent argument that reverse engineering might also be used for software piracy, the Commission considered that it could not simply allow reverse engineering without any limitation. Although existing exceptions to copyright in some Member States, such as "fair dealing/fair use", might already permit reverse engineering and, simultaneously, provide a fair scope of such permission, the application of "fair dealing/fair use" might create uncertainty and it fitted uneasily under the civil law jurisdiction. Therefore, the Commission attempted to identify the circumstances which Member States might find "fair" if they applied the "fair dealing/fair use" concept, and then codified those circumstances into a series of checks and balance, which became the conditions in Article 6 of the Software Directive.[25]

Article 6 sets many conditions which have to be met before Article 6 comes into operation. As stated above, the reason why many conditions are imposed is to minimise chances to use reverse engineering to produce pirate software as much as possible. This is also due to the pressure from SAGE which strongly objects to the legality of reverse engineering.

Article 6 sets three initial conditions with which the user has to comply in order to perform reverse engineering legally. The first condition is:

---

[22] Morrison, supra note 6, at 302.
[23] Morrison, supra note 6. She viewed that the use of the word "suffer" was probably too strong because copyright law did not grant any entitlement to revenues from the unprotected, underlying ideas in publicly distributed software products.
[24] Recital 2 and 3.
[25] B. Czarnota and R. Hart, *Legal Protection of Computer Programs in Europe: A Guide to the EC Directive* (Butterworths: London, 1991) p. 75

> [Decompilation can only be] performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorised to do so.[26]

The first condition seems to be reasonable because it protects the revenue of the copyright holder and does not encourage software piracy. Hackers or possessors of pirate copies are thus excluded from this privilege.[27] However, this provision may not be useful for programs sold at a low price which any individual can afford. But it will be extremely useful to prevent the hacker from decompiling a large system which only large companies and organisations can afford to purchase. From this consequence, it can be argued that the software manufacturer which produces an expensive system will indirectly but effectively be able to prevent the free flow of information to the public. This is because it will automatically exclude small companies which cannot afford the high price of software from creating a new derivative market.

The second condition provides that:

> The information necessary to achieve interoperability has not previously been readily available to the persons referred to in [the first condition].[28]

This condition seems impractical. In practice, when information necessary to achieve interoperability is available, the information is often incomplete or may be intentionally made incomplete, or is shortly out of date when the next version of the product is released.[29] As a result, there will be a gap in the law which the software manufacturer can use to prevent decompilation. This condition also poses a question as to why the law prohibits one from obtaining the same information by helping himself by way of decompilation when the information is readily available.[30] The economic status of the copyright holder in this situation will not be affected. Rather, the user performing reverse engineering to obtain the same information will put

---

[26] Article 6(1)(a)
[27] Czarnota and Hart, supra note 25, at 79.
[28] Article 6(1)(b)
[29] Morrison, supra note 6, at 316; Stephen Saxby, *Encyclopedia of Information Technology Law* Volume I (Sweet & Maxwell: London, 1990-2000) para. 2.367.
[30] Ibid.

himself into an economically disadvantaged position because the process of reverse engineering incurs a large expense and is time consuming. In practice, if complete information is readily available, it is unlikely that the programmer or company would invest time and money in engaging the tedious decompilation process in order to obtain the same information. Therefore, there does not seem to be any logic in making such provision.

The third condition provides:

> [Decompilation must be] confined to the parts of the original program which are necessary to achieve interoperability.[31]

It can be argued that confining the analysis to only those parts of the program concerned with interoperability is very restrictive and impractical. For example, if no documentation has been made available, it is almost impossible for the programmer to determine in advance which parts of the decompiled program contain relevant information. In this situation, the entire program must be decompiled, thereby rendering the reverse engineering activity outside the scope of this condition.[32] Furthermore, although the parts containing information have been specified, few programmers would be comfortable merely accepting this revelation without further analysis and study.[33]

There are further three limitations on the use of the information obtained through decompilation. If the user breaches one of the limitations, the act of decompilation will be illegal. The first limitation is:

> [The information must not] be used for goals other than to achieve the interoperability of the independently created computer program.[34]

---

[31] Article 6(1)(c).
[32] Morrison, supra note 6, at 316. See Saxby, supra note 29 at para. 2.369. See also the CLRC Report on Computer Software Protection, Chapter 10, p. 193. <http://www.agps.gov.au/customer/agd/clrc/clrc/word%20comp%20software%20for%20ws/indexCS.html> accessed on 14/6/99.
[33] Morrison, supra note 6.
[34] Article 6(2)(a).

It can be inferred from this limitation that decompilation can be performed only for the purpose of creating a compatible program. This limitation seems to confirm the narrow scope of the legality of decompilation, which can be seen from the wording in Article 6(1). As a consequence of this rigid limitation, questions may arise as to why the Software Directive does not allow the information to be used for other purposes, e.g. for the purpose of research or private study or for maintenance, and why it does not allow decompilation to be carried out just to discover underlying ideas and principles. This limitation seems to ignore the real purpose of reverse engineering and confuses the purpose of reverse engineering with its application.

It is submitted that the first limitation does inherently create a diversion from the fundamental principle of copyright law. That is to say, it allows the copyright holder to prevent the use of his or her ideas which underlie a computer program. This is because if the user does not use the information for the specified goal, the whole process of decompilation will be illegal even though decompilation is carried out simply to understand the ideas behind the program. The first limitation is also inconsistent with the recitals 13 and 14 which read as follows:

> Whereas, for the avoidance of doubt, it has to be made clear that only the expression of a computer program is protected and that ideas and principles which underlie any element of a program, including those which underlie its interfaces, are not protected by copyright under this Directive.[35]
>
> Where as, in accordance with this principle of copyright, to the extent that logic, algorithms and programming languages comprise ideas and principles, those ideas and principles are not protected under this Directive.[36]

Therefore, it is submitted that although this limitation attempts to restrict the use of reverse engineering to that which will not facilitate software piracy and not affect the market of the rightholder, it distorts the fundamental principle of copyright law.

The second limitation provides:

---

[35] Recital 13.
[36] Recital 14.

> [The information must not] be given to others, except when necessary for the interoperability of the independently created computer program.[37]

This condition seems reasonable since it would be unjustified to impose conditions on the person who performs decompilation while allow others access to the information which he has made public.[38] However, it can be argued that if the first person decompiling the original program meets the conditions, it is unlikely that further disclosure of the same information to public will be harmful to the copyright holder. Therefore, this limitation may be viewed unnecessary.

The third limitation states:

> [The information must not] be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright.[39]

The third limitation poses uncertainty and casts doubt on the success of the harmonisation of software protection in the European Community, as well as creating an unnecessary link to the end product. Since the third limitation forbids the information to be used for the development or production of a computer program substantially similar to the decompiled program,[40] it can be seen that the legitimacy of reverse engineering is dependent on the end product. Thus, in practice whether or not reverse engineering is legal cannot be determined until the end product has been created. It should be noted that when the first limitation is read together with the third limitation, it is clear that Article 6 relies on the end product in determining infringement. This creates a flaw in practice because the developer will be able to determine whether his practice of reverse engineering is illegal only when he has completed the project.

In addition, it is unclear whether the consideration of "substantially similar" would take into account the merger between ideas and expression doctrine. A new program

---

[37] Article 6(2)(b).
[38] Czarnota and Hart, supra note 25, at 81.
[39] Article 6(2)(c).
[40] Ibid. "Substantially similar" in the program's expression should exclude screen display because screen display is a product of a computer program, not a program itself.

which is created with the aim of interoperability needs to have exactly the same interface code in order to interoperate perfectly with the decompiled program.[41] In this circumstance, similarity between two programs cannot be avoided. Especially, if the interface code is a program itself, it is obvious that the whole interface program must be copied into the new program to ensure a complete interoperability. Therefore, if "substantially similar" is not tested against the merger doctrine, the third limitation will never be met in this circumstance and the rightholder can use this loophole to prevent the practice of reverse engineering.

The uncertainty of the third limitation can also be seen from the fact that the phrase "substantially similar" may be interpreted differently from one Member State to the others. If the courts articulate "substantially similar" to include the structure, sequence and organisation of computer programs, it is likely that decompilation to create a computer program which competes with the decompiled program is not permitted. This suggestion is supported by the ultimate safeguard found in Article 6(3) which bars any possibility of allowing decompilation 'to be used in a manner which unreasonably prejudices the right holder's legitimate interests or conflicts with a normal exploitation of the computer program'.[42] In addition, the lack of the definition of "interoperability", it may be argued, creates uncertainty as to whether it can be interpreted to include the creation of a competing, interoperable program, and whether it should include interoperability at the user interface level. From the conditions which have to be met before performing decompilation and the conditions relating to the utilisation of the information obtained through decompilation, the court is likely to interpret the term "interoperability" narrowly and allow developers to create a computer program that is merely interoperable with the decompiled program at the technical interface level.[43]

This thesis argues that because Article 6 legalises decompilation to a certain extent only and there are several conditions, which are likely to be illusory in practice, to be met before decompilation can be performed, the commercial problem (no.1) will

---

[41] Timothy S. Teter, 'Merger and the Machines: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Cases' (1993) 45 Stanford Law Review 1061, 1063-65.
[42] Article 6(3). See also Saxby, supra note 29 at para. 2.369.
[43] Davey Kent, 'Reverse Engineering of Computer Programs' (1993) 4 Australian Intellectual Property Journal 59, 61.

remain unsolved. And the practical result flowing from this legal answer is that the outsider and the consumer will still find it difficult to use reverse engineering techniques in developing an interoperable product.

2.1.2 Inability to perform corrective maintenance

The Software Directive also gives a direct answer to the corrective maintenance problem in Article 5(1), which allows the user to perform error correction without an authorisation from the rightholder. The legal reasoning underpinning this is that the EC Commission considered that a rigid application of copyright law principles to functional works in these circumstances would have the effect of granting *de facto* monopolies for services beyond the scope of the right.[44] Therefore, the exclusive rights of the copyright holder have to be subject to a limited exception.[45]

The reasoning of the Commission is consistent to the practice in the software industry. In certain circumstances, companies or organisations may wish to perform the maintenance task themselves because they may have their own programmers who have been involved in the specification, installation and testing of the software, and, consequently, they may have some knowledge of the software.[46] Therefore, they may view that by employing in-house programmers they can save the budget spent on software maintenance. If the Software Directive does not allow error correction, it would be clear that there would have been a monopoly in the after market.

Accordingly, the Commission stated in Article 5(1) that:

> In the absence of specific contractual provisions, the acts referred to in Article 4(a) and (b) shall not require authorisation by the rightholder where they are necessary for the use of the computer program by the lawful acquirer in accordance with its intended purpose, *including for error correction*. (*emphasis added*).

---

[44] Mark Powell, 'The Software Directive'.
[45] Recital 17.
[46] David Bainbridge, *Software Copyright Law* 3rd ed. (Butterworths: London, 1997) p. 239.

The wording of Article 5(1) clearly permits corrective maintenance. Although Article 5(1) does not state precisely that the act of decompilation for the purpose of error correction is permitted, it can be inferred from the wording that decompilation can be performed for this reason. This is because Article 5(1) allows the user to perform restricted acts which are necessary for the process of decompilation without imposing any stringent conditions. However, it is questionable whether such privilege should be overridden by contract.

In Recital 18, the Software Directive states its aim that the act of correcting error may not be prohibited by contract but Article 5 states conversely that the right to correct error is subject to contract. In the economic aspect, by leaving the right to correct error subject to contract, both parties will be ensured that they can secure the best interest for their businesses, provided that both parties have the same level of negotiating power. In other words, if one party has much less negotiating power than the other, such as an individual consumer, making the error correction right subject to contract may put the individual into a disadvantaged position and the law should intervene in this circumstance. In addition, if the law specifically provides that the right to correct error may not be prohibited by contract, it will encourage competition in the software maintenance industry because there will be more third party companies entering the software maintenance business.

2.2 The unsolved problems

As stated above, the Software Directive does not provide straight answers to the rest of the problems. However, it may be inferred from the relevant provisions in the Software Directive that some of these unsolved problems are already indirectly answered.

With respect to the commercial problems concerning inability to use reverse engineering to develop competing products and related products, academic opinions are diverse. Some argue that the problems are indirectly answered by Article 6.[47]

---

[47] Morrison, supra note 6, at 305-318; Gilbert-Macmillan, supra note 5, at 257-261; Andreas Wiebe, 'European Copyright Protection of Software From a German Perspective' (1993) 9 Computer Law &

Article 6 limits reverse engineering only for the purpose of interoperability. Thus, developing (by way of reverse engineering) an idea-based product, which does not require any interoperability, is not permitted. Similarly, developing a competing product may not be permitted because Article 6(3) expresses clearly that 'the provisions of [Article 6] may not be interpreted in such a way as to allow its application to be used in a manner which unreasonably *prejudices the right holder's legitimate interests or conflicts with a normal exploitation of the computer program*' (*emphasis added*). However, some argue that developing an idea-based product and competing product may be permitted under the Software Directive, although Article 6 provides unclear answers.[48] This is because the Commission indicated in its proposal that a competing product can be created through reverse engineering and in such a case it may not be connected to the reverse engineered program. Nonetheless, at least two commentators view that neither approaches are correct.[49] They argue that a more accurate statement is that a program developed by the application of Article 6 may compete with the decompiled program provided that the decompilation is done for the purpose of achieving interoperability.[50] This means that decompilation may not be carried out in order to make a competing product which is not required to interoperate in any way with the programs which are the source of the information on which it is based.

The legal reasoning underpinning this prohibition seems to be that the Commission considered that the development of computer programs required the investment of considerable human, technical and financial resources,[51] thereby, permitting reverse engineering to be used to create a competing and idea-based product (which is not an interoperable product) would be financially unjustified for the rightholder.

To evaluate whether this legal reasoning is sound, four underlying principles on which modern international system of copyright is founded have to be considered.

Practice 79, 83; and Eric Alexander Dumbill, 'EC Directive on Computer Software Protection' [1991] Computer Law & Practice 210, 212.

[48] Powell, supra note 44, at 7; M. Lehmann and T. Dreier, 'The Legal Protection of Computer Programs: Certain Aspects of the Proposal for an (EC) Council Directive' [1990] Computer Law & Practice 92, 95.

[49] Czarnota and Hart, supra note 25, at 83.

[50] Ibid.

[51] Recital 2.

The underlying principles are considered to be fourfold, namely natural law, just reward for labour, stimulus to creativity and social requirements.[52] If copyright protection is to be bestowed against the use of the decompiled program to create a competing program, the four underlying principles have to be justified.

The prohibition on the creation of a competing program can pass the natural law principle because the competing program directly attacks the integrity of the decompiled work as the competing program will be the work of the same type as that of the decompiled program. As for the "just reward for labour" principle, it can be said that the decompiled program may be directly exploited by allowing the competing product used the expression of the decompiled program. This would directly affect the remuneration of the copyright holder of the decompiled program. Furthermore, the competing program may affect the economic basis of the creator of the decompiled program for the investment required to create a new work. Therefore, stimulus to creativity is justified for such prohibition.

Although the competing program will meet the social requirement principle for the first sale, it may conversely affect the public interest at the last. This is because, without monopoly protection, not enough new products will be developed.[53] Therefore, it is reasonable to interpret the provisions of the Software Directive that such provisions do not permit reverse engineering to create a competing product. However, if the competing program, albeit the work of the same type, is not unreasonably harmful to the market of the decompiled program, it would seem no reason to interpret Article 6 in such a way to prohibit the creation of competing programs. For example, a computer game program may be considered to be the work of the same type as a decompiled game program but maybe the work is not harmful to the decompiled game program because it simply gives the customer more choices and the customer may easily purchase both of them.[54]

---

[52] Kevin Garnett, Jonathan Rayner James and Gillian Davies, *Copinger and Skone James* (Sweet & Maxwell: London, 1999) p. 29 quoting S.M. Stewart, *International Copyright and Neighbouring Rights* (Butterworths: London, 1983) p. 22 and G. Davies, *Copyright and the Public Interest* Vol. 14, IIC Studies (Weinheim, VCH VERlag, 1994) p. 10.

[53] Lawrence D. Graham and Richard O. Zerbe, Jr. 'Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure' (1996) 22 Rutgers Computer & Technology Law Journal 61, 71.

[54] The Ninth Circuit Court of Appeals' observation in *Sega v Accolade.*

It is interesting to note that if reverse engineering for the purpose of creating a competing product or an idea-based product is performed by way of "black box" reverse engineering, it is permitted under Article 5(3) which reads:

> The person having a right to use a copy of a computer program shall be entitled, without the authorisation of the rightholder, to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.

Although Article 5(3) does not state that "black box" reverse engineering is permitted, it can be inferred from the wording of this Article that "black box" reverse engineering is permitted because the acts permitted under this Article facilitate the process of "black box" reverse engineering. Because "black box" reverse engineering does not require an analysis of the actual source or object code of the reverse engineered software, the interactive process of altering the input and observing the differing outputs to determine the ideas falls within the ambit of Article 5(3).

In principle, Article 5(3) should clearly allow "black box" reverse engineering practices and should prevail over and above any contractual agreement prohibiting "black box" reverse engineering because the Article states clearly that 'the person having a right to use a copy of a computer program shall be entitled, *without the authorisation of the rightholder*, to observe, study or test the functioning of the program'. But the position is uncertain because of the wording at the end of the Article which sets an implicit condition to be met before "black box" reverse engineering can be performed. The implicit condition is in the following troublesome phrase:

> ... if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.

It seems that to perform "black box" reverse engineering legally, the user must be entitled to perform the acts of loading, displaying, running, transmitting or storing the

target program. The questions arising here are concerned with what kinds of the acts of loading, displaying, etc. a lawful user is entitled to do. Does it merely refer to those acts necessary for the use of the computer program or does it have to be interpreted according to the provisions in Article 4(a)? This may leave a loophole for the copyright holder to prohibit "black box" reverse engineering indirectly.

If this troublesome phrase refers to acts essential to the utilisation of a computer program, it would not cause any problem. The legality of "black box" reverse engineering would be complete. But if this phrase is meant to be interpreted in accordance with Article 4(a), which reads:

> ... Insofar as loading, displaying, running, transmission or storage of the computer program necessitate such reproduction, such acts shall be subject to authorisation by the rightholder.

it would eventually allow the copyright holder to prohibit "black box" reverse engineering. This is because the phrase "which he is entitled to do" at the end of Article 5(3) can be inferred that certain types of the acts of loading, displaying, running, transmitting or storing the program require a separate permission from the copyright holder. In other words, the phrase may be interpreted that if the user is not entitled to do the acts of loading, displaying, etc., he or she will not be able to assert the rights to observe, study or test the program under Article 5(3).[55] This seems to be an indirect way to impair the enforceability of Article 5(3) by using the inclusive right provided by Article 4(a), since the acts of loading, displaying running, transmitting or storing necessitate reproduction, thus requiring the copyright holder's authorisation. Consequently, the copyright holder, who is afraid of competition, would deny authorisation of all acts save for those essentially necessary for the normal use of the program.

However, the explanatory memorandum of the Software Directive shed some light on the ambiguity of such a phrase. The Commission explained that the word "reproduction" should not be confused with "replication".[56] The recreation of a

---

[55] Morrison, supra note 6, at 300.
[56] COM (88) final, at C 91/10.

program in part or in whole while running or operating the program as part of the internal processes of the computer amounts merely to "replication", not "reproduction", because no second permanent copy of the program is made during this process. Temporary copying, moving and storing may leave no trace once the operation of the machine has terminated, although parts of the program will be reproduced and stored in other parts of the memory of the computer during the operation of the program.[57] The "reproduction" or "copying" in the traditional sense does not take place in this process, unless a "back-up" copy of the program is made.

The reason why the Commission considered the act of loading of the program as one of the restricted acts was that it foresaw the future development of computer technology which would enable future programs to be contained in media which can be inserted physically into the computer, such as chips, or may be an integral part of the hardware. In this situation, a temporary copy of the program while running or operation would not be necessary.[58] Therefore, if the temporary copy of a program is considered as "replication", rather than "reproduction", the user will undoubtedly be entitled to carry out "black box" reverse engineering and the troublesome reading of such phrase can be avoided.

However, the view of differentiating between "replication" and "reproduction" may be compromised by a persuasive US authority where it was held that the creation of a copy of the computer program in RAM constituted an infringement of copyright by copying.[59] In addition, it is submitted that the view of the Commission is in contrast to the wording of Article 4(a), which clearly indicates that "reproduction" of a computer program can be the permanent or temporary one. Thus, the difference between "replication" and "reproduction" is illusory. Finally, it can be concluded that the copyright holder can still use this loophole by using a licence agreement to prevent the practice of "black box" reverse engineering, unless the troublesome phrase is to be interpreted in the common sense as mentioned above.

---

[57] Ibid.
[58] Ibid.
[59] *Mai Systems Corp. v Peak Computer Inc.*, 991 F. 2d 511, 26 (9th Cir. 1993).

The next question arising is whether the scope of Article 5(3) can be broadened to legalise the other kind of reverse engineering, i.e. decompilation (which can be used to create competing product or an idea-based product). When Article 5(3) is interpreted in conjunction with Article 4(b) – which provides that the translation, adaptation, arrangement and any other alteration of a computer program are included in the exclusive rights – decompilation is apparently not permitted. This is because the normal use of a computer program does not involve the adaptation or translation of the program to a higher level and the user is usually prohibited from doing so by the licence agreement. Moreover, Reed suggests that the wording of Article 5(3) implies that decompilation is not allowed because Article 5(3) permits observation, but not copying or adaptation.[60]

Although this thesis agrees that it can be implied that decompilation is not permitted under Article 5(3), the thesis disagrees with Reed's reason. The thesis begs to differ that observing or studying the program in order to determine the ideas and principles are the concept of reverse engineering, whereas copying and adaptation are the process involved in performing decompilation. Thus, they cannot be compared in order to explain why decompilation is not allowed as such. The correct view should be that, decompilation is not permitted since Article 5(3) merely allows an analysis only when the user performs the acts of loading, displaying, running, transmitting or adaptation, not when he or she performs or has performed the acts of converting the computer program.

In addition, Czarnota and Hart were of the opinion that because Article 5(3) permits the observation, study and testing of the *functioning* of the program and not of the *program* itself, the acts of reproduction and translation or transformation which are concerned with reverse engineering code are not applicable here.[61] Czarnota and Hart's brief explanation is sound but, it is submitted, the difference between the study of the functioning of the program and of the program itself is insignificant. The study of the program will lead eventually to the learning of how the program functions because the study of the program itself and the study of the functioning of the

---

[60] Chris Reed, 'Reverse Engineering Computer Programs without Infringing Copyright', (1991) 2 European Intellectual Property Review 47, 52-53.
[61] Czarnota and Hart, supra note 25, at 70.

program arguably have the same aim. Nonetheless, it can be said that the study of the functioning of the program does not require a conversion of object code to source code but the study of the program does require such a conversion which is the necessary process of decompilation. Therefore, if the Software Directive had intended to extend Article 5(3) to cover decompilation, it would have permitted clearly the study of the program itself. The intention of the Software Directive to limit the scope of Article 5(3) to "black box" reverse engineering can be seen from the recital as follows:

> Whereas a person having a right to use a computer program should not be prevented from performing acts necessary to observe, study or test the functioning of the program, provided that these acts do not infringe the copyright in the program.

This recital confirms that if the observation, study or testing has to involve the infringing acts, such observation, study or testing will not be permitted. It seems that this Recital takes into account the Commission's opinion in the Explanatory Memorandum about the difference between "reproduction" and "replication", otherwise the observation during the normal use of a computer program ("black box" reverse engineering) would not be permitted.

To summarise, the commercial problems derived from the lack of the legitimate "black box" reverse engineering may not be solved by the Software Directive because of the uncertainty of the application of Article 5(3) in legalising "black box" reverse engineering. Although it may be argued that Article 5(3) perfectly legalises the practice of "black box" reverse engineering, its legitimacy will not solve all of the commercial problems described in Chapter two as the commercial problems largely stem from the need for the approval of decompilation.

With respect to other kinds of commercial problems relating to inability to perform certain types of software maintenance, i.e. perfective, adaptive and preventive, the Software Directive does not provide any answers. It cannot be inferred that these types of maintenance are permitted under the privilege "to correct errors" because the programmer usually performs these types of maintenance even when the system does

not have a malfunction or "bug". Nor can they be assumed to be permitted under Article 6 because none of them involves interoperability with other programs. Thus, the error correction (Article 5(1)) and decompilation (Article 6) exceptions do not come into play. However, it may be argued that these types of maintenance may come within the scope of the activities necessary for the use of the computer program, which do not require authorisation by the rightholder.[62] Nonetheless, if each type of maintenance is considered carefully, it can be seen that only adaptive maintenance may be qualified for the exception provided by Article 5(1). This is because adaptive maintenance may be seen as actions necessary for the use of the system.[63]

On the other hand, perfective and preventive maintenance are arguably outside the scope of the exception provided in Article 5(1) since enhancement of the performance of the system may be viewed unnecessary, unless it can prove that the want of enhancement will greatly affected the stability of the business. Likewise, actions taken to make subsequent maintenance more efficient and reliable may be considered not essential for the normal use of the computer program. Therefore, authorisation of the rightholder is still required.

Owing to this deficiency of the Software Directive in coping with commercial problems, it is submitted that the European Commission should consider the taking of a new initiative to amend the Software Directive to cope with these problems. Although the European Commission stated in its 2000 report on the implementation and effects of the Software Directive[64] that it does not see fit to undertake any initiatives to amend the Directive at this stage,[65] the commercial problems shown in this thesis and the interested circles' suggestion that there is a need for clarifying certain aspects of the Directive[66] have now warranted such an initiative, at least in respect of the right to do reverse engineering. It may be noted that the study carried out by the Commission in preparing this report was completed in 1997. The data

---

[62] Article 5(1).
[63] However, it is doubted that the provisions will be interpreted to allow adapting for a change of the supporting environment, network or hardware platform.
[64] Report from the Commission to the Council, the European Parliament and the Economic and Social Committee on the implementation and effects of Directive 91/250/EEC on the legal protection of computer programs, COM(2000) 199 final.
[65] Ibid., at p. 21.
[66] Ibid., at p. 17.

illustrated in that report, e.g. an increase in employment and harmonisation for employee-created computer programs,[67] could be said to be out-of-date. Moreover, there is no indication of an increase of competition since the implementation of the Directive. Thus, it can be said that it is now the time to open the floodgate of debate on this issue again. The approach suggested in the next chapter can be used as a model for solving these problems.

3. UK COPYRIGHT LAW

The United Kingdom chose to implement the Software Directive by way of a Statutory Instrument under powers contained in the European Communities Act 1972. This is because it considered that the CDPA 1988 already conformed substantially to the provisions of the Software Directive. Therefore, an amendment to the CDPA 1988 was inserted by the Copyright (Computer Programs) Regulations 1992 (hereinafter referred to as the Copyright Regulations) in order to avoid any prolongation which could occur from the changes made by Parliament if the draft were to be presented in the form of a new bill.

The issues arising concern whether the same deficiency and uncertainty in the Software Directive are carried over into the CDPA 1988 and whether the wording of the CDPA 1988 which paraphrases that of the Software Directive deviates from the intention of the Software Directive. Article 5 and 6 of the Software Directive are transformed into section 50A, 50B and 50C.[68] It should be noted that the discussion will focus on section 50B as it is an equivalent of Article 6 which permits decompilation.

---

[67] Ibid., at p. 16.

[68] The manner of incorporating the decompilation right was to remove decompilation from the ambit of the fair dealing provisions relating to research and private study contained in s. 29 and then introduce it as a new permitted act in s. 50B (Diane Rowland and Elizabeth Macdonald, *Information Technology Law* (Cavendish Publishing Limited: London, 1997) p. 61).

3.1 The direct answers

As one of the Member States, the United Kingdom has also provided straight legal answers to two commercial problems, i.e. inability to develop interoperable products and inability to perform corrective maintenance. However, it is argued that the scope of the legal answers is slightly different from that of the Software Directive because of the use of paraphrasing. The legal reasoning of the use of paraphrasing is that Mr. Edward Leigh, the Parliamentary Under-Secretary of State for Technology, believed that the use of paraphrasing would clarify uncertainty in the Software Directive and still adhere rigidly to the spirit of the Software Directive.[69] Hence, this part considers whether the CDPA 1988 (as amended by the Copyright (Computer Programs) Regulations 1992) provides more certainty and adheres to the spirit of the Software Directive as claimed by Mr. Leigh, and evaluates the practical results flowing from the UK legal answers whether they are different from those of the Software Directive.

3.1.1 Inability to develop interoperable products

At first glance, it appears that the person who can perform reverse engineering under section 50B is defined as a "lawful user" whereas Article 6 uses the term "licensee". A "lawful user" might have a slightly wider scope than that of the term "licensee" as a "lawful user" includes persons who have right to use the program under a licence *or otherwise*.[70] It may be argued that because of the word "otherwise", anyone would be classified as a lawful user since he or she has a *statutory* right to decompile a computer program. However, it is doubted that the court would interpret the word "lawful user" as such because it would contrast with the intent of the Software Directive. Some commentators argue, on the other hand, that by not adding after the word "lawful user" the phrase "or on his behalf by a person authorised to do so", it seems that only the lawful user himself has a right to decompile. He may not employ a third party to do the decompilation task for him.[71] It is submitted that this view is

---

[69] Parliamentary Debates: First Standing Committee on Statutory Instruments, &c., 'Draft Copyright (Computer Programs) regulations 1992' Thursday 10 December 1992 (HMSO, 1992) p. 8-9.
[70] Section 50A(2) provides: 'For the purposes of this section and sections 50B and 50C a person is a lawful user of a computer program if (whether under a licence to do any acts restricted by the copyright in the program or *otherwise*), he has a right to use the program. *(emphasis added)*
[71] Andreas Gunther and Ulrich Wuermeling, 'The German Implementation of the EC Directive on Software Protection' [1993] 9 The Computer Law and Security Report 176, 178.

rather narrow and seems not to take into account of the effect of the word "otherwise" as mentioned earlier.

As it paraphrases Article 6, section 50B uses the words "convert" and "copy" instead of "translation" and "reproduction" as used in Article 6 respectively. Although the use of synonym poses no harm in this situation, the description of the condition along with the word "copy" may create inconsistency. That is to say, because section 50B(1)(b) provides that "it is not an infringement ... *incidentally* in the course of so converting the program, to copy it", it may be argued that a manual reproduction of the code onto a paper by the programmer during converting the program will render the whole process of reverse engineering illegal. This is because the manual reproduction will create a permanent copy, as opposed to incidental copy created in RAM which will be deleted once the computer is switched off. In addition, the use of the phrases "a low level language" and "a higher level language",[72] which are not found in the Software Directive, may create inconsistency. The conversion of a binary object code program into hexadecimal code, something which is commonly known as performing a "hex dump", does not produce a higher level language. Therefore, such a conversion may not be permitted under section 50B of the CDPA 1988 but it would be within the exception as expressed in Article 6 of the Software Directive.[73]

The way the conditions in Article 6 of the Software Directive are rearranged may create inconsistency and a lack of clarity. Such inconsistency is obvious in the wordings of section 50B(3)(a). Section 50B(3)(a) provides that the conditions for reverse engineering are not met 'if the lawful user has *readily available* to him the information necessary to achieve the permitted objective', whereas Article 6(1)(b) refers to "the information necessary to achieve interoperability has not *previously been readily available*". It is argued that current availability and previous availability

---

[72] Section 50B(1):
It is not an infringement of copyright for a lawful user of a copy of a computer program expressed in *a low level language* – (a) to convert it into a version expressed in *a higher level language. (emphasis added)*

[73] David Bainbridge, *Intellectual Property 4th ed.* (Financial Times/Pitman Publishing: London, 1999) p. 218-219.

are two different concepts.[74] Under section 50B(3)(a), a copyright holder may easily design a circumstance surrounding the grant of a software licence in such a way as to ensure that this condition is not met.[75] If the copyright holder states in the software licence that, upon request, the licensor would supply source and interface specifications, section 50B(3)(a) would apply.[76] But if this scenario is considered under Article 6(1)(b), it would be that the lawful user can perform reverse engineering legally because the information has not *previously* been readily available. Moreover, if the information is subject to an unreasonable fee, it is arguable whether such information is still *readily* available.

The only uncertainty of Article 6 that has been clarified in Article 50B is that the information is prohibited from being passed to a third party. Section 50B(3)(c) provides that the conditions are not met if the lawful user 'supplies the information obtained by decompilation to any person to whom it is not necessary to supply it in order to achieve the permitted objective'. This situation is uncertain under Article 6 as Article 6(2)(b) arguably allows a third party to use the information. This is because Article 6(2)(b) does not define the person who wishes to create the independent program in the phrase "except when necessary for the interoperability of the independently created computer program".

From the analysis above, although some uncertainty in the Software Directive is clarified by the CDPA 1988, the difference in the wording of Article 6 and that of section 50B will create further uncertainty. Therefore, the position of reverse engineering for the purpose of interoperability in the UK is still unclear. As a result, the commercial problem concerning inability to develop an interoperable product is unsolved in practice.

---

[74] Simon Chalton, 'Implementation of the Software Directive in the United Kingdom: The Effects of the Copyright (Computer Programs) Regulations 1992' [1993] 9 The Computer Law and Security Report 115, 118.
[75] Michael Rhodes, 'Prohibiting the Decompilation of Software' (1993) 9 Computer Law & Practice 1.
[76] Ibid.

3.1.2 Inability to perform corrective maintenance

Section 50C(2) of the CDPA 1988 clearly states that error correction can be done without infringing copyright. But this exception can also be cancelled out by contract. The issue arising here is whether or not the UK common law principle of non-derogation from grant will apply. In the UK, there exists a principle of non-derogation from grant developed in the House of Lords case of *British Leyland v Armstrong*.[77] In this case, the plaintiff was a car manufacturer. The plaintiff granted licences to other companies to make spare parts for its car. The defendant refused to obtain such a licence to make exhaust pipes for the plaintiff's car. The plaintiff brought an action against the defendant on the ground of indirect infringement of the copyright subsisting in the drawings of the exhaust pipes.

The House of Lords held that the defendant had infringed the plaintiff's copyright in the drawings of the exhaust pipes but that the copyright could not be enforced. The reason given was that car owners had an inherent right to repair their cars as economically as possible and therefore, they had a right to have access to a free market in spare parts. The plaintiff could not interfere with that right by controlling the market by means of its copyright.

Many scholars view that there would seem to be no logical reason why the non-derogation from grant principle could not also be extended to an agreement for the acquisition of computer software.[78] The principle would mean, in this context, that the lawful acquirer of computer software would be able to decompile, modify and recompile the program code without infringing the copyright even if these actions were expressly prohibited by the agreement.[79] This would also mean that the user can approach third parties with a view to maintaining his program where the supplier is prepared to provide this service.[80]

However, the non-derogation from grant in the British Leyland case should be treated with some caution. In the light of a recent Privy Council case of *Canon Kabushiki*

---

[77] *British Leyland Motor Corp Ltd. v Armstrong Patents Co Ltd.* [1986] 2 WLR 400.
[78] Bainbridge, *Software Copyright Law*, supra note 46, at 245.
[79] Ibid.
[80] David Bainbridge, *Software Licensing* (Central Law Publishing: Birmingham, 1995) p. 32.

Kaisha v Green Cartridge Company (Hong Kong) Limited,[81] it was held that 'once one departed from the case in which the unfairness to the customer and the anti-competitive nature of the monopoly was as plain and obvious as in *British Leyland* the jurisprudential and economic basis for the doctrine became extremely fragile'.[82] In this case, Green Cartridge attempted to defend its alleged infringement of copyright of engineering drawings relating to the cartridge used with the plaintiff's printers and photocopiers. Green Cartridge claimed that the making and supply of the new cartridges fell within the "spare parts" exception recognised in the British Leyland case. The Privy Council rejected the Green Cartridge's argument. Their Lordships distinguished this case from the British Leyland on several grounds. One of their Lordships reasons was that unlike the case of the replacement of a car exhaust which was relatively small in relation to the capital and other running costs of a car, the cost of cartridges was relatively important as a proportion of the lifetime cost of the photocopiers or laser printer.[83] By the same token, the cost of software maintenance accounts for 50-90 percent of the lifetime costs of the system.[84] Thus, it can be argued that the British Leyland exception may not apply to the software market for this reason.

3.2 The unsolved problems

Like the Software Directive, the CDPA 1988 does not provide direct answers to the rest of the addressed problems. Therefore, the answers must be implied from the relevant provisions of the CDPA 1988.

Under the CDPA 1988, the problem concerning inability to develop competing products may be treated differently due to the wording of section 50B. Section 50B(2)(a) provides that:

> It is necessary to decompile the program to obtain the information necessary to create an independent program which can be operated with the program decompiled or with another program.

---

[81] *Canon Kabushiki Kaisha v Green Cartridge Company (Hong Kong) Limited* [1997] FSR 817.
[82] Ibid., at 818.
[83] Ibid.

From this wording, the lawful user is allowed to create two types of programs, namely, one which can interoperate with the decompiled program and one which can interoperate with another program which has not been decompiled. It is, therefore, questionable whether the lawful user can create a new program that can interoperate and, at the same time, compete with the program decompile or another program, and whether he or she can create a new program that can interoperate with another program but cannot interoperate with the decompiled program (and compete with it). It is clear from Article 6 that the licensee cannot use the information to create a competing program because it obviously states in paragraph (3) that the provisions 'may not be interpreted in such a way as to allow its application to be used in a manner which unreasonably prejudices the right holder's legitimate interests or conflicts with a normal exploitation of the computer program'. However, paragraph (3) has not been implemented in the CDPA 1988, thereby leaving some doubts for the software industry as to whether such practice is permitted. In an absence of such provision, it is submitted that the court might interpret in favour of competition policy, viz. allowing the creation of a competing product, provided that the new product has to be interoperable in certain ways with the decompiled program.

With respect to the commercial problem concerning inability to create idea-based products, the result would be similar to that of the Software Directive. Because the idea-based product does not have to be interoperable with the decompiled program, section 50B will not apply. The prohibition of the creation of idea-based products is confirmed by section 29 which excludes from fair dealing for the purpose of research and private study the acts necessary for the decompilation process. Section 29 provides:

> It is not fair dealing–
>
> (a) to convert a computer program expressed in a low level language into a version expressed in a higher level language, or
>
> (b) incidentally in the course of so converting the program to copy it,
>
> (these acts being permitted if done in accordance with section 50B (decompilation)).

---

[84] See Chapter two, supra, at page 10.

However, it can be inferred that the practice of "black box" reverse engineering is permitted and it cannot be prohibited by contract because it is stated clearly in section 296A(1)(c) that the use of any device or means to observe, study or test the functioning of the program in order to understand the ideas and principles which underlie any element of the program cannot be prohibited by contract.

In respect of the maintenance problems (perfective, adaptive and preventive), none of them is permitted under the CDPA 1988. Even adaptive maintenance which may be permitted under the Software Directive does not survive under the provisions of the CDPA 1988. Section 50C restricts copying or adaptation for the purpose of correcting errors only. Therefore, adaptive maintenance which normally does not involve error correction but rather involves platform immigration is unlikely to be permitted. To conclude, it can be seen that the application of reverse engineering to software maintenance in the UK is rather limited and the commercial problems concerning the inability to perform the other three types of software maintenance will continue to exist in the UK. Although the CDPA 1988 attempts to provide provisions which create more certainty and which adhere to the spirit of the Software Directive, the use of paraphrasing still posts a certain level of uncertainty and carries over the impracticality of conditions and limitation in Article 6 of the Software Directive to the CDPA 1988. Therefore, this thesis argues that current UK copyright law cannot solve the commercial problems as explained above.

4. US COPYRIGHT LAW

4.1 The legal response

The United States has provided a different approach to tackle the commercial problems, i.e. using the doctrine of "fair use"[85] to legalise reverse engineering. In fact, the US approach is originally derived from the traditional doctrine of "fair

---

[85] 17 U.S.C § 107.

dealing" in the United Kingdom and this doctrine was once referred to in the discussion of how to legalise reverse engineering in the United Kingdom.[86] Therefore, it is worthwhile to study the US approach and to make a comparison as well as to analyse the differences, which may have occurred because of the passage of time, between the doctrine of "fair use" and "fair dealing". This is to consider whether the US approach gives a satisfactory solution. If it does so, it may be suggested that the UK, in the light of the uncertainty found in section 50B, should change its approach to follow that of the US. If, however, the result reveals that the US approach is not satisfactory either, this should confirm that the current legality of reverse engineering needs to be reconsidered and another solution will be required. My submission is that the US approach is not satisfactory and this leads to my suggestion in the form of the proposed framework to be considered in the next chapter.

The US Copyright Act 1976 does not provide a clear-cut answer to the problems as the Software Directive and the CDPA 1988 do. The US Copyright Act 1976 instead leaves the issues for the court to decide within the flexible scope of the common law "fair use" doctrine. Until now, there are two Courts of Appeals cases which address some of the problems raised in this thesis, and give legal answers to them. However, the legal answers given are slightly different, which may lead to two different practical results.

The first legal answer was given by the Federal Circuit in *Atari v Nintendo*[87] and the second answer given by the Ninth Circuit in *Sega v Accolade.*[88] The Federal Circuit was of the opinion that reverse engineering object code to discern the unprotectable ideas in a computer program should be qualified as a fair use. By virtue of this opinion, all of the commercial problems raised by this thesis should be solved easily because the Federal Circuit did not link the legitimacy of reverse engineering to its end application. However, because the judgment lacked critical analysis of the statutory factors provided in the fair use provision (to be discussed later) the Federal Circuit's opinion cannot be said to be an indisputable approach of the United States.

---

[86] ECIS (UK Implementation Working Group), 'Implementation in UK Legislation of EC Software Directive – The ECIS Response' [1992] 8 The Computer Law and Security Report 11, 13.
[87] *Atari v Nintendo* 975 F. 2d 832 (Fed. Cir. 1992).

The second answer, which receives mixed reception from both academia and courts in subsequent cases, solves merely the first commercial problem (inability to develop interoperable products). However, it can arguably be extended to solve other commercial problems, provided that the broadening of this legal answer still meets the criteria given by the four statutory factors. What the other commercial problems are, that may be solved under this second legal answer, will be analysed.

4.2 Evaluation of legal reasoning underpinning and practical results

The thesis now turns to analyse the legal reasoning underpinning each legal answer. By way of the traditional analysis of case law, it will start with the fact of the case. In *Atari v Nintendo*, *Atari* wanted to enter the home video game market by supplying game cartridges compatible with the *Nintendo*'s video game console. However, because of the security system of the *Nintendo*'s video game which prevented the game console from accepting unauthorised game cartridges, *Atari* had to find the way to circumvent that security system. The security system of *Nintendo* was the 10NES program. This program was embedded both in the *Nintendo*'s game cartridge and in the video game console, called the Nintendo Entertainment System (NES), to prevent the NES from accepting unauthorised game cartridges.[89] In order to unlock the NES, *Atari* had to create a program functioning similarly to the 10NES. *Atari*, therefore, reverse engineered the 10NES program. But *Atari* failed in the first attempt. It could not decompile the code sufficiently to replicate the 10NES. Then, by falsely obtaining a copy of the 10NES source code program from the Copyright Office, it used the source code program to correct errors in the translated version of the 10NES object code, thereby being able to develop its own program, called the Rabbit program. By imitating the functions of the 10NES, the Rabbit program could generate signals similar to those of the 10NES and, thus, unlock the NES. *Nintendo* sought a preliminary injunction prohibiting *Atari*'s exploitation of *Nintendo*'s

---

[88] *Sega v Accolade* 977 F. 2d 1510 (9th Cir. 1992).

[89] The 10NES program works in the following manner. Both the NES console and authorised game cartridges contain microprocessors or chips programmed with the 10NES. The console contains a "master chip" or "lock". Authorised game cartridges contain a "slave chip" or "key". When a user inserts an authorised cartridge into a console, the slave chip in effect unlocks the console; the console

copyrighted computer program. *Nintendo* alleged, *inter alia*, that 'copies of the 10NES program made by *Atari* in the reverse engineering process infringed the 10NES copyright'.[90] The District Court was of the opinion that reverse engineering was copyright infringement.[91] Therefore, it granted an injunction. *Atari* appealed.

The Court of Appeals, Federal Circuit, disagreed with the District Court's opinion on the reverse engineering issue. The Federal Circuit viewed reverse engineering object code to discern the unprotectable ideas in a computer program as a fair use.[92] It reasoned that because in principle the Copyright Act permits an individual in rightful possession of a copy of a work to undertake necessary efforts to understand the work's idea, processes, and methods of operation, an author cannot acquire patent-like protection by putting an idea, process, or method of operation in an unintelligible format and asserting copyright infringement against those who try to understand that idea, process, or method of operation.[93] The Federal Circuit pointed out that this permission appears in the fair use exception to copyright exclusivity,[94] which exempts from copyright protection reproductions for criticism, comment or research.

The reasoning of the Federal Circuit seems very consistent with, and in fact it is an application of, the fundamental principle of copyright and the traditional concept of fair use which were clarified more than a century ago by the US Supreme Court in *Baker v Selden*.[95] The Supreme Court stated that:

> The very object of publishing a book on science or the useful arts is to communicate to the world the useful knowledge which it contains. But this object would be frustrated if the knowledge could not be used without incurring the guilt of piracy of the book.

---

detects a coded message and accepts the game cartridge. When a user inserts an unauthorised cartridge, the console detects no unlocking message and refuses to operate the cartridge.

[90] *Atari v Nintendo*, supra note 87, at 840.

[91] *Atari Games Corp. v Nintendo of America, Inc.*, 18 U.S.P.Q.2d 1935 (N.D.Cal. 1991).

[92] *Atari v Nintendo*, supra note 87, at 843-844.

[93] *Atari v Nintendo*, supra note 87, at 842.

[94] The Court of Appeals quoted section 107 of the Copyright Act as follows: section 107 states that 'fair use of copyrighted work, including such use by reproduction in copies ... for purposes such criticism, comment, new reporting, teaching ... scholarship or research' is not infringement.

[95] *Baker v Selden* 101 U.S. 99 (1879).

Although this case concerned a traditional literary work, the principle derived from this case can be applied squarely within the context of software reverse engineering. It, therefore, can be said that it would greatly undermine the primary objective of copyright if the knowledge and ideas in the computer program could not be used without infringing copyright in the computer program.

In the *Atari* case, the Federal Circuit supported reproduction for research because this activity permits public understanding and dissemination of the ideas, processes, and methods of operation in a work. It suggested further that 'the copyright holder has a property interest in preventing others from reaping the fruits of his labour, not in preventing authors and thinkers of the future from making use of, or building upon, his advances'.[96] Based on this suggestion, the Federal Circuit concluded that:

> Where the infringement is small in relation to the new work created, the fair user is profiting largely from his own creative efforts rather than free-riding on another's work. A prohibition on all copying whatsoever would stifle the free flow of ideas without serving any legitimate interest of the copyright holder.

The restriction on the exercise of the right to copy is rational and reflects the fact that all reproductions of the work are not within the exclusive domain of the copyright holder; some are in the public domain. Therefore, any individual may reproduce a copyrighted work for a fair use; the copyright holder does not possess the exclusive right to such a use.[97] The only aspect that would not justify fair use is the commercial exploitation of protected expression. The court clearly said that 'the fair use reproductions of a computer program must not exceed what is necessary to understand the unprotected elements of the work'.[98] This seems to be well justified because the concept of reverse engineering is entirely concerned with the understanding of the unprotected elements of the work. Anything beyond that point falls within the scope of forward engineering. Thus, it can be said that in general the Federal Circuit's principle supports the practice of reverse engineering.

---

[96] *Atari v Nintendo*, supra note 87, at 843.
[97] *Sony Corp. v Universal City Studios, Inc.* 464 U.S. 417, 433 (1984f).
[98] *Atari v Nintendo*, supra note 87, at 843.

However, although the suggestion of the Federal Circuit is persuasive and can be applied to allow reverse engineering, the lack of a critical analysis of the criteria provided in the fair use provision may be a hindrance to its application. It is rather disappointing that the Federal Circuit in *Atari v Nintendo* did not examine in detail the four statutory factors provided in the "fair use" provision.[99] The only factor referred to briefly was the second factor.[100] The Federal Circuit was of the opinion that 'when the nature of a work requires intermediate copying to understand the ideas and processes in a copyrighted work, that nature supports a fair use for intermediate copying'.[101] Although this brief analysis is reasonable, the lack of analysis of the other three factors may lead to an argument that the reasoning for the legality of reverse engineering in this case may not be adequate. The outcome of the case might be different if all factors were analysed in full. However, it seemed that the reason why the Federal Circuit did not analyse those factors in detail was that the defendant obtained a purloined copy from the Copyright Office, and this amounted to possessing an unauthorised copy of the program.[102] The Federal Circuit, thus, used this fact to deny the application of fair use, stating that 'knowing exploitation of purloined manuscript was not compatible with "good faith" and "fair dealings" underpinnings of fair use doctrine'.[103] Thus, it can be seen that the commercial problems raised at the beginning of this chapter would not readily be solved by the doctrine of fair use.

Moreover, a crucial flaw in this judgment, which made the application of fair use in doubt, was that the Court considered and applied fair use on the wrong material.
The Federal Circuit erred in applying fair use to the source code obtained from the Copyright Office. In fact, the Court should have considered fair use in relation to the 10NES object code program because the issue in this case was whether the intermediate reproduction of the object code (as a result of the process of reverse engineering) was a fair use. Therefore, the object code should be the centre of the Federal Circuit's determination of whether it should refuse the application of fair use to authorise the infringing activity. If *Atari* had received the 10NES object code program by an unlawful means from the Copyright Office and performed reverse

---

[99] 17 U.S.C § 107.
[100] § 107(2).
[101] *Atari v Nintendo*, supra note 87, at 843.
[102] John A. William, 'Can Reverse Engineering of Software Ever Be Fair Use? Application of Cambell's "Transformative Use" Concept' (1996) 71 Washington Law Review 255, 263 at n. 75.

engineering on such object code, it should have been prohibited from asserting the fair use exception. What happened in this case, however, was that the 10NES object code program, which was reverse engineered, was legally acquired by *Atari*. Furthermore, the 10NES program falsely obtained from the Copyright Office was the source code, which was not reverse engineered by *Atari* but rather was merely used to assist the understanding of the 10NES object code program which *Atari* had in it possession before. Hence, the application and rejection of fair use in this case were not appropriate. Therefore, it is submitted that the principle derived from this case cannot be used to answer the question of whether or not it is appropriate to use fair use to legalise reverse engineering. If there were to be an answer to it, it is likely that fair use cannot legalise reverse engineering as the compliance with the statutory criteria of fair use make it impossible to do so as discussed above.

A question arising at this point is whether or not it is possible for the UK's fair dealing principle, which does not have the same statutory criteria, to be interpreted in the way suggested by the Federal Circuit mentioned above. The answer to this question is that there may be a difficulty in adopting such an interpretation due to the difference in the nature of fair dealing and fair use.

In the past, the question of what amounted to fair dealing frequently arose in determining whether or not the use which had been made of the plaintiff's work was sufficient to constitute infringement.[104] As such, this question was often not distinguished from the issue of whether a substantial part of the plaintiff's work had been taken.[105] Generally speaking, when a part of a work has been taken, if it is not a substantial part, the taking is presumably fair. On the other hand, if a substantial part has been copied this would be an unfair use, hence copyright infringement. Such an interpretation of fair dealing was followed in *Independent Television Publications Ltd v Time Out Ltd.*[106] The Court in this case held that if the whole or substantial part of the work had been taken a defence of fair dealing was unlikely to succeed.

---

[103] *Atari v Nintendo*, supra note 87, at 843.
[104] Kevin Garnett, Jonathan James and Gillian Davies, *Copinger and Skone James on Copyright 14th ed.* (Sweet & Maxwell: London, 1999) p. 496 quoting *Wilkins v Aikin* (1810) 17 Vesey 422; *Scott v Stanford* (1867) L.R. 3 Eq. 718; *Bradbury v Hotten* (1872) L.R. 8 Ex. 1; *Smith v Chato* (1874) 31 L.T. 77.
[105] Ibid.
[106] *Independent Television Publications Ltd v Time Out Ltd.* [1984] FSR 64, 75.

However, the interpretation of fair dealing as such was subsequently questioned by leading scholars.[107] It was suggested that the question of fair dealing should be raised only where a substantial part or the whole work had been taken.[108] This meant that where an infringement had already occurred the defence of fair dealing would be raised to excuse the defendant from such infringement. As the current case law and the CDPA 1988 suggest the aforementioned, it seems unlikely to interpret fair dealing in the same way as the Federal Circuit did for fair use, even though there is no statutory criteria as such for fair dealing.

This makes it necessary to consider whether or not a greater detailed analysis of fair use in *Sega v Accolade* where all listed factors in section 107 were examined, would assist in legalising reverse engineering.

Following *Atari v Nintendo*, the Court of Appeals, Ninth Circuit, produced a deeper analysis on each element of the fair use exception. Its analysis and application of fair use arguably provides for more flexibility than the provisions under the Software Directive and the CDPA 1988. Therefore, the issues arising are: whether the Ninth Circuit Court of Appeals' analysis of fair use is justified, whether the application of fair use can solve all the commercial problems described in Chapter two, and whether the US approach of using fair use to legalise reverse engineering should be followed by the UK courts. Keeping in line with the traditional way of analysing the case, the facts and arguments of the case will be presented first.

The fact in this case was similar to that of *Atari v Nintendo*. In brief, *Accolade* developed its own game cartridges to be compatible with the *Sega*'s game console by way of reverse engineering. *Accolade* performed reverse engineering on both *Sega*'s game cartridges and game console in order to obtain the common interface specification for the game console. In developing its own game cartridges, *Accolade* did not copy *Sega*'s program, except for the interface specification which was necessary for compatibility. *Sega* filed suit against *Accolade*, alleging, *inter alia*, that

---

[107] Copinger and Skone James on Copyright, 13th ed., para 10.6; Paul Sumpter, *Copyright and Design* (Butterworths: Auckland, 1996); Sam Ricketson, *Intellectual Property: Cases, Materials and Commentary* (Butterworths, Sydney, 1994).

Accolade infringed its copyright in video game programs. The District Court held in favour of *Sega*. *Accolade* appealed.

Accolade raised four defences[109] to the copyright infringement claim, only one of which was considered having merit, namely the defence of fair use. The Court of Appeals, Ninth Circuit, held that reverse engineering copyrighted object code fell squarely within the concept of fair use because it was the only method available to access the unprotected elements of the program and because *Accolade* had a legitimate interest in gaining such access.[110] In so holding, the Ninth Circuit went on to analyse in detail the four factors listed in section 107 to support its conclusion that 'where there is good reason for studying or examining the unprotected aspects of a copyrighted computer program, [reverse engineering] for purposes of such study or examination constitutes a fair use'.[111]

Section 107 provides that in determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include –

(1) The purpose and character of the use, including whether such use is of a commercial nature or is for non-profit educational purposes;
(2) The nature of the copyrighted work;
(3) The amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
(4) The effect of the use upon the potential market for or value of the copyrighted work. The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.

---

[108] *Copinger and Skone James on Copyright 14th ed.*, supra note 104, at para 9-22
[109] First, the intermediate or interim copying did not infringe the exclusive rights granted to the copyrights owners unless the end product of the copying is substantially similar to the original product. Second, because the copyright law does not protect the ideas and functional concepts, disassembly of the object code so as to understand those ideas and functional concepts is legal. Third, because the Copyright Act 1976 (USA) allows a lawful user to load the program into a computer, disassembly is authorised. Fourth, disassembly for the purpose of understanding the ideas and functional concepts is fair use under § 107.
[110] *Sega v Accolade*, supra note 88, at 1520.
[111] Ibid.

In this part, the reasoning of the Ninth Circuit on each factor will be assessed against criticism on the judgment made by well-known scholars in the field. They are William, Clapes and Lai (when this thesis refers to William hereinafter, it includes Clapes and Lai who support William's approach).[112]

The first factor

The Ninth Circuit considered that the reproduction at issue was merely an intermediate one. Therefore, any commercial exploitation was indirect or derivative.[113] This finding obviously conflicts with the ruling of the Supreme Court in *Harper & Row v Nation*[114] where it was said that commercial purpose tended to weigh against a finding of fair use because:

> the crux of the profit/non-profit distinction is not whether the sole motive of the use is monetary gain but whether the user stands to profit from exploitation of the copyrighted material without paying the customary price.[115]

The Ninth Circuit denied that this was always the case. It pointed out that the presumption of unfair use that arose in such a case could be rebutted by the characteristics of a particular commercial use such as the present case, and the commercial nature of a use was a matter of degree, not an absolute.[116] Moreover, it found that *Accolade* did not avoid paying a fee for use of those procedures because *Accolade* purchased both the game cartridges and consoles. Taking these facts into account, the Ninth Circuit found it sufficient to conclude that the presumption of

---

[112] *See* John A. William, 'Can Reverse Engineering of Software Ever Be Fair Use? Application of Cambell's "Transformative Use" Concept' (1996) 71 Washington Law Review 255; Stanley Lai, 'Recent Developments in Copyright Protection and Software Reverse Engineering in Singapore: A Triumph for the Ultra-protectionists?' [1997] 9 European Intellectual Property Review 525; Anthony L. Clapes, 'Decompilation & European Protection of Computer Programs: A Review of Legal Protection of Computer Programs in Europe: A Guide to the EC Directive by Bridget Czarnota & Robert J. Hart' (1994) 20 Rutgers Computer & Technology Law Journal 329. This thesis refers chiefly to William because he gives the most comprehensive details to support his argument.

[113] *Sega v Accolade*, supra note 88, at 1522.

[114] *Harper & Row Publishers, Inc v Nation Enterprises* 471 U.S. 539 (1985). This case involved the interpretation of fair use. The respondent in this case took the unpublished manuscript of the appellant and published it under the respondent's article, thereby resulting in the cancellation of Time Magazine of contract which was agreed to pay a sum of money in return for the copyright in this manuscript. The Supreme Court referred to §107 which the respondent raised as a defence of copyright infringement.

[115] *Harper & Row v Nation*, ibid., at 562.

[116] *Sega v Accolade*, supra note 88, at 1522.

unfairness was overcome because the 'direct use of the copyrighted material was simply to study the functional requirements for compatibility so that it could modify its games to be compatible with the console'[117] and this amounted essentially to non-exploitative purpose.[118] The finding of non-exploitative purpose apart, the Ninth Circuit was of the opinion that public consideration of competition policy might prevail upon copyright protection. It noted that although the use of a work might allow the infringer to gain commercially, the courts might permit such a use if it serves a public interest which the Copyright Act was intended to promote.[119]

William argued that the reasoning was unpersuasive because '*Accolade*'s admitted purpose behind its infringement was to allow *Accolade* to directly compete against *Sega* videogames *(sic)*, which should have been sufficient to convince the court that *Accolade*'s ultimate aim was direct commercial exploitation'.[120] He argued further that the Ninth Circuit failed to address the commercial benefits *Accolade* had intended to secure through its research of *Sega*'s code, and failed to consider that although the inherent research nature of reverse engineering could always be characterised as serving a legitimate purpose, the purpose of *Accolade*'s reverse engineering was undeniably commercial.[121]

It is submitted that the reasoning of the Ninth Circuit in this respect is sound but William's arguments are incorrect. The reasoning of the Ninth Circuit is in line with the concept of reverse engineering. The concept of reverse engineering reveals that the nature of reverse engineering lies obviously in research. Instead, the commercial application of reverse engineering is the inherent one because any commercial application is not in the process of reverse engineering; it is in the process of forward engineering.[122] Therefore, the Ninth Circuit's emphasis on the obvious "research" nature of reverse engineering, not on the inherent commercial application, is perfectly correct.

---

[117] Ibid..
[118] *Sega v Accolade*, supra note 88, at 1522-1523.
[119] *Sega v Accolade*, supra note 88, at 1523. See also *Feist Publications, Inc. v Rural Telephone Service Co.*, 499 U.S. 340 (1991).
[120] William, supra note 112 at 267.
[121] Ibid., at 268.
[122] Full discussion of this concept is presented in Chapter four.

William's argument that *Accolade*'s aim was direct commercial exploitation is also a misreading of the facts of the case. The facts stated by the Ninth Circuit made it clear that *Accolade* reverse engineered *Sega*'s games in order to discover interface specification only. It did not incorporate any copyrighted element of *Sega*'s games in its final products. This was considered by the Ninth Circuit as "non-commercial exploitation". William obviously failed to distinguish between infringement the result of which could be perceived from the final product (hence, directly compete with the original product) and infringement which simply assisted the final product to enter the market (hence, not directly compete). This was why he confused himself in making the argument that *Accolade*'s aim was direct commercial exploitation.

Furthermore, William's final argument regarding the Ninth Circuit's first factor analysis was that the Ninth Circuit failed to appreciate the investment that *Sega* put into the creation of its video game system. Thus, he argued, the Ninth Circuit did not consider carefully enough as to whether or not *Sega*'s copyright needed protection against the fair use doctrine to justly reward *Sega*'s extensive efforts.[123] Again, William erred in the fact of the case. There was no evidence that *Accolade* sought to avoid spending its own investment and creative effort in creating its own game cartridges. Indeed, most of the games that *Accolade* released for use with the *Sega*'s game console were originally developed for other hardware systems.[124] Moreover, *Accolade* did not copy the game console or creating a competing game console which the user normally needed only one. Therefore, the investment of *Sega* was still secured. It must be said, the Ninth Circuit's reasoning in this aspect is persuasive.

The second factor

The Ninth Circuit applied the principle of "different levels of protection". That is, works of fiction receive greater protection than works that have strong factual elements[125] or works that have strong functional elements.[126] Applying this principle to a function work like a computer program, the Ninth Circuit gave a lower degree of copyright protection. Because *Sega*'s video game programs contained unprotected

---

[123] William, supra note 112, at 268.
[124] *Sega v Accolade*, supra note 88, at 1522.
[125] *Rosemont Enterprises, Inc. v Random House, Inc.* 366 F.2d 303, 307 (2nd Cir. 1966).
[126] *Baker v Selden* 101 U.S. 99.

aspects that could not be examined without copying, they could not enjoy the same degree of protection as traditional literary works. Otherwise, as the Ninth Circuit expressed, the owner of the copyright would gain a *de facto* monopoly over the functional aspects of this work – aspects that were expressly denied copyright protection by Congress.

William argued that this analysis was also questionable because the Ninth Circuit, in holding that software deserved a lower degree of protection than other literary work, contradicted Congress's intent in making software copyrightable. He said that 'Congress intended to protect software programs without regard to how ideas or functions were embedded within these works' and that 'copyright does not impose a duty on copyright owners to allow unrestrained access to their works'.[127] Therefore, in emphasising the difference between computer programs and other literary works, the Ninth Circuit 'carves a special niche in copyright law that is not authorised by statute'.[128]

Again, William appeared to err in giving his argument. The Ninth Circuit's application of the "different level of protection" principle was not just established in this case; it was established more than a century ago and reiterated several times by the Supreme Court.[129] Moreover, the Congress's intent to protect computer programs under copyright law does not mean that ideas and functions of computer programs will be protected as can be seen from the provisions of section 102(b) of the Copyright Act 1976 (USA), which excludes from protection any idea, procedure, method of operation, concept, principle, etc.

William referred to the report of the National Commission on New Technological Uses of Copyrighted Works (CONTU) when interpreting the Congress's intent as such. This can be seen as a clear mistake in reading the CONTU Report because the CONTU Report clearly stated that 'when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given

---

[127] Ibid., at 269.
[128] Ibid., at 269.
[129] *Baker v Selden* 101 U.S. 99; *Feist Publications, Inc. v Rural Telephone Service Co.*, 499 U.S. 340 (1991).

task, their later use by another will not amount to infringement'.[130] Therefore, although the CONTU Report did not directly state that "copyright owners had to allow unrestrained access to their work", it should be so because copyright did not provide such a wide exclusive right to the copyright owner. The reading of the CONTU Report as suggested by this thesis is consistent with the previous authorities, which Congress, indeed, attempted to codify.

It may be noted that the "different level of protection" principle can apply only to US copyright law because US copyright law has never developed to protect functional work like that of the UK.[131] This is important when considering the context of copyright infringement which will be discussed in the next chapter.

The third factor

The Ninth Circuit was of the opinion that this factor weighed in favour of *Sega* because copying in the process of reverse engineering is "whole-sale" copying. However, by following the authority in *Sony v Universal City Studios*[132] and *Hustler v Moral Majority*[133], it considered that copying the entire work did not preclude a finding of a fair use and that the factor was of very little weight in this case.

William stated that the Ninth Circuit's analysis in respect of this factor was the least persuasive because firstly, the Ninth Circuit's reliance on the *Sony* case was misplaced and, secondly, it contradicted itself in its rationale. He said that the Ninth Circuit failed to recognise that the Supreme Court in the *Sony* case (at page 421) restricted its holding to cases concerning "time-shifting".[134] Therefore, the Ninth Circuit's application of fair use from the *Sony* case to the present case was misplaced. From William's suggestion, the application of fair use as such was prohibited by previous authority. But, a close reading of the case reveals that nothing in page 421 is said to limit the application of fair use to the time-shifting case only. In fact, the

---

[130] CONTU Report at 20.
[131] *British Leyland Motor Corp Ltd v Armstrong Patents Co Ltd* [1986] 2 WLR 400; *LB Plastics Ltd v Swish Products Ltd* [1979] RPC 551 compare with *Baker v Selden* (1880) 101 US 99; *Feist Publications Inc v Rural Telephone Service Co. Inc.* (1991) S Ct 1282.
[132] *Sony Corp., v Universal City Studios, Inc.*, 464 U.S. 417 (1984).
[133] *Hustler Magazine, Inc. v Moral Majority, Inc.*, 769 F.2d 1148, 1152 (9th Cir. 1986).
[134] Ibid., at 270.

Supreme Court does not want to expand the copyright privilege beyond the limits of the grants authorised by Congress because it would enlarge the scope of the copyright holder' statutory monopolies.[135] Hence, William's argument was not convincing.

However, William was correct with regard to his argument that the Ninth Circuit contradicted itself when it stated that this factor was of very little weight because it was only intermediate copying.[136] The Ninth Circuit made a contradictory statement because it stated at the beginning that 'the Copyright Act does not distinguish between unauthorised copies of a copyrighted work on the basis of what stage of the alleged infringer's work the unauthorised copies represent' but it said, when evaluating this factor, that "whole-sale" copying in this case weighted so little because the copying was in an intermediate process.[137] Therefore, the reasoning of the Ninth Circuit on this factor is unsound.

The fourth factor

The Ninth Circuit found itself constrained by the established interpretations of the fourth factor that 'if the challenged use should become widespread, it would adversely affect the potential market for the copyrighted work'[138] and that 'if a use effectively usurps the market for the copyrighted work by supplanting that work, such a use is dispositive'[139]. However, the Ninth Circuit distinguished this case in that the use of the copyrighted work in this case merely to enable the infringer to enter the market for works of the same type, and *Accolade*'s games would not significantly affect the market for *Sega*'s games since the video game users normally purchased more than one game.

William argued that it was obvious that *Accolade*'s games would directly compete with *Sega*'s games and would directly affect the market. His observation of the video game market was opposite to that of the Ninth Circuit. He noted that typically most consumers could not afford to, nor would want to, buy two similar video games.

---

[135] *Sony v Universal City Studio*, supra note 132, at 421.
[136] Ibid., at 270.
[137] *Sega v Accolade*, supra note 88, at 1518.
[138] *Sony v Universal City Studio*, supra note 132, at 451.
[139] *Harper & Row v Nation*, supra note 114, at 567-69.

Circuit has recently confirmed its approach again in *Sony v Connectix*,[144] it was merely a decision on the appeal from a preliminary injunction which, it is submitted, did not carry the same authority as a decision on the case which had been trialled. Moreover, in *Sony v Connectix*, the Ninth Circuit did not address the question of "indirect copying" and "derivative work" which could have been found in Connectix's final product. These questions, if they had been answered, could have changed the outcome of the case. As the case was remanded, the final judgment is yet to be seen.

Although the detailed judgment will be given in due course, some aspects of the latest judgment are worth considering. As the Ninth Circuit allowed reverse engineering on the basis that the final product was transformative and that it was in a different market, it is unlikely that reverse engineering which would create a product that directly competes with the original product will be allowed. Furthermore, as the Ninth Circuit did not consider other uses of reverse engineering which may not lead to the creation of a final product, i.e. those in software maintenance, their legal status is still unclear. Therefore, it is submitted that the commercial problems still exist, despite the application of the doctrine of fair use.

On the question of whether or not the US interpretation of fair use will or should be analogously adopted in to the UK for the interpretation of fair dealing, it is submitted that it is unlikely. This is because there is no statutory factor in the fair dealing provisions which guides the court in determining whether or not the use at issue is fair dealing with a work. Furthermore, the UK concept of use – which is considered unfair, for example, where the part copied is so valuable as to be calculated to undermine the original work, as by diminishing its sales or profits[145] or superseding its objects;[146] or to save the copyist an undue amount of the trouble the original author had been at[147] – may not have any formal bearing on the interpretation of the CDPA 1988.[148] Indeed, it is arguable that "fair use" has been an unjustly neglected topic in modern UK copyright law, while in the United States the concept, derived from the

---

[144] *Sony v Connectix* 53 USPQ.2d 1702, 1709 (9th Cir. 2000).
[145] *Bradbury v Hotten* (1872) LR 8 Exch 1, 6.
[146] *Scott v Stanford* (1876) LR 3 Eq 718.
[147] *Jarrold v Houlston* (1857) 3 K & J 708, 717.

Hence, *Sega* was likely to suffer considerable revenue losses.[140] Moreover, its revenue from the licensing scheme would decrease dramatically as more companies would follow the practice of reverse engineering in the same way to avoid the licensing fee.[141]

This thesis argues that the observations of both William and the Ninth Circuit are not convincing. This is for at least two reasons. First, there was no statistic to support their observation. This, it can be inferred, they simply assumed from their experience. Second, they did not separate the categories of consumers. For example, the age of consumers, which implies their working status, may directly affect the purchasing power. Therefore, the impact on *Sega*'s market was merely speculative. However, if this case were involved with a highly functional computer program such as a word processing or spreadsheet program, it could have been suggested with more certainty that consumers might have bought one program only because any one of them could perform the same function.[142]

Nonetheless, the argument regarding the decrease in the revenue derived from the licensing scheme of *Sega* was convincing. William was correct in stating that the Ninth Circuit failed to address the potential harm to *Sega*'s existing licensing market as the other licensees might follow *Accolade* by using reverse engineering instead of paying the licence fee.[143] It can be said that the Ninth Circuit failed to answer the Supreme Court inquiry of whether, if the use should become widespread, it would adversely affect the potential market for the copyrighted work. Therefore, considering the above analysis, it could not be suggested that the fourth factor weighs in favour of *Accolade*.

Applying the equitable rule of reason, it can be concluded that it is very controversial that reverse engineering would be permitted under the fair use doctrine in the future. At the present time, there seems to be more and more academic opinions against the unorthodox reasoning of the Ninth Circuit in *Sega v Accolade*. Although the Ninth

---

[140] Lai, supra note 112, at 532.
[141] Ibid.
[142] Unlike a word processing or spreadsheet program, the differences of user interface are indeed the crucial factor which attracts customers to buy more than one game of the same type.
[143] William, supra note 112, at 272.

Circuit has recently confirmed its approach again in *Sony v Connectix*,[144] it was merely a decision on the appeal from a preliminary injunction which, it is submitted, did not carry the same authority as a decision on the case which had been trialled. Moreover, in *Sony v Connectix*, the Ninth Circuit did not address the question of "indirect copying" and "derivative work" which could have been found in Connectix's final product. These questions, if they had been answered, could have changed the outcome of the case. As the case was remanded, the final judgment is yet to be seen.

Although the detailed judgment will be given in due course, some aspects of the latest judgment are worth considering. As the Ninth Circuit allowed reverse engineering on the basis that the final product was transformative and that it was in a different market, it is unlikely that reverse engineering which would create a product that directly competes with the original product will be allowed. Furthermore, as the Ninth Circuit did not consider other uses of reverse engineering which may not lead to the creation of a final product, i.e. those in software maintenance, their legal status is still unclear. Therefore, it is submitted that the commercial problems still exist, despite the application of the doctrine of fair use.

On the question of whether or not the US interpretation of fair use will or should be analogously adopted in to the UK for the interpretation of fair dealing, it is submitted that it is unlikely. This is because there is no statutory factor in the fair dealing provisions which guides the court in determining whether or not the use at issue is fair dealing with a work. Furthermore, the UK concept of use – which is considered unfair, for example, where the part copied is so valuable as to be calculated to undermine the original work, as by diminishing its sales or profits[145] or superseding its objects;[146] or to save the copyist an undue amount of the trouble the original author had been at[147] – may not have any formal bearing on the interpretation of the CDPA 1988.[148] Indeed, it is arguable that "fair use" has been an unjustly neglected topic in modern UK copyright law, while in the United States the concept, derived from the

---

[144] *Sony v Connectix* 53 USPQ.2d 1702, 1709 (9th Cir. 2000).
[145] *Bradbury v Hotten* (1872) LR 8 Exch 1, 6.
[146] *Scott v Stanford* (1876) LR 3 Eq 718.
[147] *Jarrold v Houlston* (1857) 3 K & J 708, 717.

4. SECOND STEP SUGGESTION: APPROVE THE PRACTICE OF REVERSE ENGINEERING AND FORWARD ENGINEERING

Accordingly, this thesis proposes that the law should approve the practice of reverse engineering. That is, to surrender copyright protection of computer programs against reverse engineering. The approval of reverse engineering will be consistent with the aim and principle of copyright law. Also, because the thesis argues that preventing the practice of reverse engineering is not the real issue, the approval of reverse engineering will support this suggestion by moving the focus of the debate onto the real issue, i.e. how to prevent unfair use of information (and, in the context of copyright law, unfair use of expression). Moreover, by approving reverse engineering it will abolish the wrong use of copyright as a mechanism to protect ideas and information instead of using the law of confidence or trade secret law. Therefore, it will free up ideas and stimulate greater encouragement of competition in the software market.

As mentioned above, the approval of reverse engineering will move the focus of the debate onto the use of information. It is submitted that the use of information is in the process of forward engineering because it is the stage where the programmer uses the information to create a new product. Therefore, the law may create regulations to control the use of information as such. In other words, the emphasis of legal regulations should shift away from reverse engineering to forward engineering. The change of emphasis will free up research and stimulate the study in pure computer science which are usually relied on the practice of reverse engineering. The determination of infringement in the process of forward engineering is also possible because if the source code or documentation discovered in the process of reverse engineering is forward engineered (compiled) into the end product, it will amount to copying of the original program. Especially for the common law fair use principle, the court will not have difficulty in finding that the act of forward engineering the source code of the original program to create the end product will be a direct exploitation of a copyrighted work.[17]

---

[17] *See Sega v Accolade* and William, ibid. When compare between reverse and forward engineering, it can be seen clearly that forward engineering to create a similar program is a direct exploitation of the original program.

However, this thesis speculates that the focus on developing regulations of forward engineering may not be the best solution in terms of enforcement in practice. This is because the plaintiff may find it difficult to prove the act of forward engineering, as the act of forward engineering is usually performed secretly. Moreover, the defendant can alter his project at any time in order to exclude the information of the original program from being added into the new product. So, the determination of infringement of forward engineering cannot be certain until the end product has been created. Owing to this problem, it can be seen that although the concept of controlling the use of information is advisable in theory, the determination of infringement which has to link with the end product will make the regulations of forward engineering inappropriate in practice. As a result, this thesis proposes that the best way is to determine infringement at the end product by comparing the end product with the original product.

In conclusion, neither reverse engineering nor forward engineering should be an infringement of copyright. Such infringement should be exclusively determined by a comparison of the finished product with the original product. Comparison of the two end products will create a more open creative environment for the exploitation of ideas, and stimulate greater encouragement of competition in the software market.

5. THIRD STEP SUGGESTION: CONCENTRATE ON DEVELOPING THE TEST FOR "SUBSTANTIAL SIMILARITY" FOR THE JAVA PROGRAMMING LANGUAGE

The advent of the Java programming language has attracted interests of the computer industry because of its ability to deliver programs across the Internet. The advent of Java has resulted in the creation of new types of computer programs, which can run on the Internet, such as animation programs. More importantly, it has also advanced the object-oriented programming technology. For these reasons, Java has become

generally known as a computer language for the future.[18] Due to its significant feature, i.e. object-oriented, an impact on the legal status of computer programs written in Java cannot be avoided.

The advancement of programming technology created by Java is that the programmer can design a system as being composed of a number of objects that provide and request services from each other in order to accomplish an overall task.[19] In other words, a computer program written in Java will comprise many small programs interacting with one another, and these small programs can be obtained from the standard Java library of classes or existing Java programs. Therefore, the nature of creating a new program will swiftly move from writing text and graph from scratch to compiling finished objects. Therefore, a Java program is more or less analogous to a compilation or functioning database. Nevertheless, this object-oriented nature does not prevent the programmer from writing his or her own objects.

The potential change in the method of creating a computer program will lead to the need for reverse engineering to discover existing objects and make use of them to create a new program. If the legitimacy of reverse engineering is still uncertain or the scope of the legality of reverse engineering is very limited, it will be undeniably difficult to make full use of the Java programming technology. Therefore, the thesis' proposal to surrender copyright protection against the practice of reverse engineering does support the advancement of computer technology.

However, an effect of the thesis' proposal that any infringement should be determined from the final product, is expected. The effect is that the scope of copyright protection bestowed on Java programs may decrease due to the object-oriented nature of Java. Modern computer programs written in Java will share common objects readily available from the Java class library. In the most extreme case, a Java program may consist wholly of existing objects, let alone the selecting and arranging skill of the programmer to be presented in the program. Therefore, it may be arguable whether Java programs can meet the minimum criteria for copyright protection and

---

[18] Roger Garside and John Mariani, *Java: First Contact* (Cambridge: Course Technology, 1998) p. xxv.
[19] Ibid.

whether Java programs which are built from the objects of an existing program will infringe that existing program. These issues are very important in the course of determining infringement by comparing the allegedly infringed product and the allegedly infringing product. It is submitted that the legal standard or the current system of determining infringement should be reconsidered in order to make it appropriate for Java programs. The discussion on how the UK court should determine infringement by comparison of the end products in the light of new computer technology will be discussed below.

Determining infringement by comparing the end products is not a simple task. Under copyright law, to constitute a copyright infringement the alleged infringing work must be "substantially similar" to the original work. Therefore, the issue arising here is what can be said to constitute "substantially similar". Tests for determining substantial similarity have been developed by courts in both the United States and the United Kingdom as well as in other common law countries. However, it appears that the test, known as "the three-step test", developed by a US Court of Appeals, has been recognised world-wide and is considered by the UK courts. Therefore, this thesis seeks to examine:

i. whether this test represents good authority;
ii. whether the UK's adoption of this test with a slight change in one principle will create confusion of the test; and
iii. whether the reasoning of the UK court in rejecting this test will represent good law.

As this test was actually created in 1992, this thesis also seeks to examine whether this test is still applicable to computer programs created by the Java computer programming language.

If the analysis of the test and the analysis of the adoption and rejection of the test by the UK court show that the test is not appropriate for current computer technology, the thesis will suggest a suitable test for substantial similarity based on UK copyright law. In examining the applicability of this test, the thesis attempts to demonstrate

relationships among "look and feel", an idea/expression dichotomy and substantial similarity.

As the first stage of this analysis, an illustration of the process of proving copyright infringement is an appropriate starting point for the purpose of the present discussion. In determining whether copying constitutes infringement, the courts have to compare the copyrighted and allegedly infringing works to ascertain whether or not there are objective similarities between these two works and then to decide whether similarities which may be found to have occurred by copying, constitute a substantial part of the copyrighted work.[20] Traditionally, to prove copyright infringement by copying, the plaintiff has to show that the defendant did copy the plaintiff's work.[21] The plaintiff may infer such copying by showing that the defendant had access to the copyrighted program and that the allegedly infringing program is *substantially similar* to the copyrighted program. Where the program code has been copied literally, there seems to be less trouble for the courts to determine copyright infringement. In contrast, where there is no similarity in the literal elements of the programs, but only similarity in their overall structure and organisation, the courts will encounter a difficult issue of whether a mere similarity in the overall structure of the program will constitute copyright infringement. The underlying question of this issue is whether copyright extends to protect non-literal elements of a computer program e.g. the structure and organisation of the algorithm and subroutines.[22] The discussion of case law below will answer this question.

---

[20] *John Richardson v Flanders* [1993] FSR 497, 515; section 16 of the CDPA 1988.

[21] Frequently, in software copyright infringement cases it is hardly possible to prove such copying through direct evidence *Rot Greeting Cards v United Card Co.*, 429 F.2d 1106, 1110 (9th Cir. 1970).

[22] Non-literal elements of computer programs, e.g. structure and organisation, are sometimes referred to as "look and feel" (*see* Linda G. Morrison, 'The EC Directive on the Legal Protection of Computer Programs: Does It Leave Room for Reverse Engineering Beyond the Need for Interoperability?', 25 Vanderbilt Journal of Transnational Law 293-332, at n. 21). However, there is no exact parameter of what is meant by the phrase "look and feel". Some commentators defines "look and feel" to include the screen display, the user's interface and structure or organisation of the program code (*see* Pamela Samuelson, Robert Glushko and Linda Morrison). Some commentators even distinguish between the "look" and the "feel"; the "look" of a program includes its demonstrative audiovisual elements, its screen displays, visible portions of the user interface and other visual and aural elements of output produced by the program, the "feel" of a program means the dynamic, operational flow of the program, its keystrokes and other means for invoking functions and the general recognisable style of the operation the program presents to the user (*see* David L. Hayes, 'A Comprehensive Current Analysis of Software "Look and Feel" Protection', <http://www.fenwick.com/pub/lookfeel.html> accessed on 22 July 1998). Hayes also described the "feel" as the non-literal structure, sequence and organisation of a program – both its static modular structure and organisation, and its dynamic operational sequence of

In the United States, it has long been accepted as a general matter that copyright protection extends beyond a literary work's strictly textual form to its non-literal components as stated by Judge Hand in *Nichols v Universal Pictures Co.*[23]:

> It is of course essential to any protection of literary property, whether at common law or under the statute, that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations. That has never been the law, but, as soon as literal appropriation ceases to be the test, the whole matter is necessarily at large, so that, as was recently said by a distinguished judge, the decisions cannot help much in any new case ... When plays are concerned the plagiarist may excise a separate scene ...; or he may appropriate part of the dialogue ... Then the question is whether the part so taken is "substantial" and therefore not a "fair use" of the copyrighted work; it is the same question as arises in the case of any other copyrighted work ... But when the plagiarist does not take out a block *in situ*, but an abstract of the whole, decision is more troublesome.[24]

Therefore, if the fundamental essence or structure of one work is duplicated in another, it will constitute copyright infringement.[25] The position of copyright law in the United Kingdom appears to be the same, for the UK courts adopted this persuasive authority in both *John Richardson v Flanders*[26] and *Ibcos v Barclays*.[27]

A more difficult question is to what extent copyright protection extends to the non-literal element of a computer program. The following case law shows the development of the court's thinking in defining the scope for such protection. This will lead to the discussion of whether the latest development represents a good authority for the UK.

---

program control flow – and the "feel" has sometimes been used to encompass other non-visible elements such as file formats, data structures, commands, and system calls.

[23] *Nichols v Universal Pictures Co.*45 F2d. 119 (2d Cir. 1930).

[24] Ibid., at 121. It is suggested by Harris that where plagiarism is an appropriation of ideas, without the appropriation of the actual expression of those ideas, it is not a violation of copyright since copyright does not protect ideas (Lesley Ellen Harris, *Digital Property: Currency of the 21st Century* (McGraw-Hill Ryerson Limited: Toronto, 1998) p. 161.

[25] 3 Nimmer, para 13.03 [A][1].

[26] *John Richardson Computers Ltd. v Flanders* [1993] FSR 497.

[27] *Ibcos Computers Ltd. v Barclays Mercantile Highland Finance Ltd.* [1994] FSR 275.

In the United States, in answering this question the court usually relies on a fundamental principle of copyright law, that is, copyright does not protect an idea, but only the expression of the idea. A line between ideas and expression in the context of computer programs was once drawn in the broadest sense by the Third Circuit Court of Appeals in *Whelan v Jaslow*. The Third Circuit was of the opinion that the idea of a computer program was the end sought to be achieved by the computer program. This means that the idea was the purpose or ultimate function of that computer program. Accordingly, everything else that was not necessary to that purpose or function would be part of the expression of the idea.

However, the *Whelan* approach is not considered by commentators and subsequent courts as good authority because it is seen as an outdated appreciation of computer science. Professor Nimmer criticises that 'the crucial flaw in Whelan's reasoning is that it assumes that only one "idea" ... underlies any computer program, and that once a separable idea can be identified, everything else must be expression'.[28] This criticism is not based on the program's ultimate purpose, but based on the fact that a computer program's ultimate purpose is the composite result of interacting subroutines and indeed each subroutine is itself a computer program. Thus, when the Second Circuit Court of Appeals in *Computer Associates v Altai*[29] considered the dichotomy between ideas and expression, the Second Circuit expressed that it could not be erroneous to conclude that each subroutine or the small computer program had its own "idea"; Whelan's general formulation that a program's overall purpose could amount to the program's idea was, therefore, descriptively inadequate.[30]

After rejecting the *Whelan* approach, the Second Circuit created a test which helped the court to distinguish ideas from expression and, more specifically, to determine whether the non-literal elements of two computer programs were substantially similar. As mentioned previously, this test is known as the three-step test which consists of a) abstraction, b) filtration and c) comparison. It is this test that the thesis seeks to analyse in order to answer whether this test can cater for the new Java technology.

---

[28] Infra note 29, at 705; cited the opinion of Professor Nimmer, 3 Nimmer s13.03(F), at 13-62.34.
[29] *Computer Associates International Inc. v Altai Inc* 982 F.2d 693 (2nd Cir. 1992).
[30] *Computer Associates v Altai*, supra note 29, at 705.

According to the three-step test, the court begins the procedure by breaking down the copyrighted program into its constituent parts. Then, the court examines each part to sift out the idea from expression and also to distinguish protectable expression from the expression which, by policy reason, should not be given copyright protection. At the end of this stage, the court will be able to find a kernel of creative expression. Then, at the final step, the court will compare the protectable expression with the allegedly infringing program in order to determine whether the allegedly infringing program is substantially similar to the protectable element of the copyrighted program so as to warrant a finding of infringement.[31]

Of the three steps, the second step, filtration, is of the most significant because it is the stage where the court applies copyright doctrines which limit the scope of copyright protection. There are three doctrines which the Second Circuit employed to filter out non-protectable elements, namely the doctrine of merger between ideas and expression, the "scenes a` faire" doctrine and the doctrine that copyright gives no protection to elements taken from the public domain. The thesis now turns to consider each doctrine in detail to answer whether each doctrine can fit comfortably with a computer program created by Java. If it cannot fit so, it is submitted that a newly designed test suggested by this thesis is needed.

a. The merger doctrine

Under this doctrine, the court will consider whether a particular inclusion of any element is dictated by considerations of efficiency. This doctrine prevents the copyright holder from monopolising an idea which can be expressed in a limited number of ways. In the *Altai* case, the Second Circuit viewed that in practice the programmer usually attempted to write a program in the way that it could run efficiently and meet the user's needs in the most efficient manner. That is, to derive 'the most logical proof or [to formulate] the most succinct mathematical computation'.[32] Hence, in reality there may be only a few ways or limited file structures available to achieve a specific task, whereas in theory there might be a

---

[31] *See* Ian J Lloyd, *Information Technology Law 2nd ed.* (Butterworths: London, 1997) p. 343-346.
[32] *Computer Associates v Altai*, supra note 29, at 708.

number of ways to effectuate a particular function. The Second Circuit thus sets a standard to determine whether the merger doctrine is applied as follows:

> It follows that in order to determine whether the merger doctrine precludes copyright protection to an aspect of a program's structure that is so oriented, a court must inquire whether the use of this particular set of modules is necessary efficiently to implement that part of the program's process being implemented. If the answer is yes, then the expression represented by the programmer's choice of a specific module or group of modules has merged with their underlying idea and is unprotected.[33]

The Second Circuit noted that several programmers who work independently may design the identical structure used in the allegedly infringed work because of the efficiency constraint. This will result in no copyright infringement and the similarity cannot be used as a reference to copying.

However, it can be argued that this doctrine as well as the Second Circuit's application of this doctrine may not be applied appropriately to a Java computer program. This is because in writing a Java program the structure of a Java program is not the emphasised point owing to the nature of the Java programming language. As the nature of Java is object-oriented, it is uncommon to write a Java program using the structured and procedure-based programming techniques that were developed in the early 1970s and that were referred to by the Second Circuit in applying the merger doctrine.[34] Instead of having a particular structure of computation for a set of tasks, a Java program operates by using *method calls* which would send a message to the relevant object responsible for the performing a set of related tasks. Therefore, it cannot be determined whether the Java program's structure in question represents the most efficient computation which any programmer is likely to come up with that structure.

b. The "scene a` faire" doctrine

---

[33] Ibid. The merger doctrine in effect follows from the idea/expression concept; see Sterling, infra note 68 at 252.
[34] Cornell and Horstmann, supra note 8, at 92.

This doctrine precludes copyright protection from elements dictated by external factors. Under the scenes a` faire doctrine, expression which becomes a standard for a particular work is not copyrightable. The Second Circuit applied this doctrine to computer programs by referring to the opinion of Professor Nimmer, who pointed out that 'in many instances it [was] virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques'.[35] Nimmer summarised five external factors which circumscribed the programmer's freedom of design choice, namely:

1) the mechanical specifications of the computer on which a particular program is intended to run;
2) compatibility requirements of other programs with which a program is designed to operate in conjunction;
3) computer manufacturers' design standards;
4) demands of the industry being serviced; and
5) widely accepted programming practices within the computer industry.[36]

In relation to the Java computer program, these five factors may not all apply. Because Java source code can be compiled to "byte code", an intermediate form between source code and object code, the programmer does not have to conform to the mechanical specification of the computer and computer manufacturers' design standards. Thus, factor 1) and 3) become irrelevant in considering the scene a` faire doctrine. While the result of the exclusion of the first and third factors is an expansion of the scope of protectable expression, the effect of the fifth factor is that most of the elements of a Java program may be precluded from receiving copyright protection. This is because the "reuse" technique has become acceptable in the software engineering practice. Java programs may consist of many objects which are standard for a set of specific tasks. Hence, it can be seen that there would be a discrepancy if the court were to apply these five factors to a Java program.

c. Public domain

---

[35] 3 Nimmer s 13.03 [F][3], at 13-65.

Under this doctrine, elements taken from the public domain may not obtain copyright protection. The Second Circuit sets a rigid rule that there is no reason to make an exception to this rule for elements of a computer program that has entered the public domain by virtue of freely accessible program exchanges and the like or for elements that are common place in the computer industry.[37] The Court opined that such elements must be filtered out from the copyrighted program before the final procedure of substantial similarity analysis, i.e. comparison, takes place.

It is submitted that this rigid rule would render most Java programs unprotectable because many objects in a Java program can be obtained from the Java library or from existing programs. Moreover, the practice of reverse engineering on Java programs readily available on the Internet increases the level of reusability of objects. This casts some doubt on the appropriateness of subsistence of copyright in Java computer programs. Also because Java programs are primarily designed to run on the web page, this is equivalent to putting Java programs in the public domain. The subsequent Java program which assembles objects from different Java programs on the web page will receive no copyright protection as the element in the new program will be comprised of elements taken from the public domain.

From the analysis of the doctrines mentioned above, it can be seen that the three-step test may not be the most appropriate test for substantial similarity of computer programs created by the Java programming language. Therefore, it can be said that a new method of determining an infringement may be needed. However, before suggesting a new method, it is necessary to consider whether or not the modification made by a UK court to the three-step test will make the test suitable for Java programs, hence no need of a new test.

As the UK High court in *John Richardson v Flanders* once adopted this test with a slight change, the thesis will consider whether or not such a change would create a confusion and inconsistency with the original test established in *Computer Associates v Altai*. The analysis of the *Richardson* case will show that the way in which the UK court adopted the Altai's test is not appropriate and a new way of determining

---

[36] Ibid., at 13—66-71, cf. *Altai*, at 709.

copyright infringement is needed. The new way of determining an infringement will be illustrated at the end of this chapter.

In *John Richardson v Flanders*, Ferris J. adopted both the ideas/expression dichotomy doctrine and the three-step test into the United Kingdom. He expressed that although this doctrine was established in the United States[38] and was enshrined in the statute in the United States, it was recognised and applied in English law too.[39] In addition, Ferris J was of the opinion that the three-step test could be readily adopted into UK copyright law as 'nothing in any English decision [conflicted] with the general approach adopted in the *Computer Associates v Altai* case'.[40]

Nonetheless, Ferris J expressed his reservation about adopting the concept and process of determining infringement via the three-step test. This then becomes the issue to be discussed. Instead of dissecting the copyrighted work to seek the core of protectable expression as suggested by the Second Circuit, Ferris J was of the opinion that 'an English court [would] first decide whether the plaintiff's program as a whole [was] entitled to copyright and then decide whether any similarity attributable to copying which [was] to be found in the defendant's program amount[ed] to the copying of a substantial part of the plaintiff's program [as a whole]'.[41] Then, in assessing substantiality, the court would apply the three-step test.[42] This was because Ferris J felt obliged to follow the language used in the CDPA 1988 and the approach taken by a previous English authority, *Ladbroke (Football) Ltd. v William Hill (Football) Ltd.*[43] Section 16(3) of the CDPA 1988 provides:

> References in this Part to the doing of an act restricted by the copyright in a work are to the doing of it –
> (a) in relation to the work as a whole or any substantial part of it,...

---

[37] *Computer Associates v Altai*, supra note 29, at 709.
[38] See *Baker v Selden* 101 U.S. 99 (1879).
[39] *John Richardson v Flanders*, supra note 26, at 523. Nonetheless, he realised that there were still dangers inherent in too extensive a reliance on it. This was discussed in Laddie, Prescott and Vitoria, *The Modern Law of Copyright* (1980) at paras. 2-50 to 2-55.
[40] *John Richardson v Flanders*, supra note 26, at 526-27.
[41] *John Richardson v Flanders*, supra note 26, at 527.
[42] *John Richardson v Flanders*, supra note 26, at 500, 527.
[43] [1964] 1 WLR 273.

The speech of Lord Pearce in *Ladbroke (Football) Ltd. v William Hill (Football) Ltd.* was also cited as authority:

> In deciding therefore whether a work in the nature of a compilation is original, it is wrong to start by considering individual parts of it apart from the whole, as the appellants in their argument sought to do. For many compilations have nothing original in their parts, yet the sum total of the compilation may be original. (see, for instance, the case of *Palgrave's Golden Treasury* referred to by the Privy Council in *Macmillan & Co. Ltd. v K. & J. Cooper.*).
>
> In such cases the courts have looked to see whether the compilation of the unoriginal material called for work or skill or expense. If it did, it is entitled to be considered original and to be protected against those who wish to steal the fruits of the work or skill or expense by copying it without taking the trouble to compile it themselves.[44]

Therefore, the issues arising are:

i. Whether <u>not dissecting</u> the work at the beginning will make the three-step procedure difficult to apply and cause confusion to the test itself;
ii. Whether comparing the infringing work with the copyrighted work as a whole is correct; and
iii. Whether it is appropriate to apply the three-step test to the part which is said to have been copied, rather than to the copyrighted work as a whole.

With regard to the first issue, the purpose of the three-step test is to distinguish ideas from expression at each level of abstraction in order to find out elements that should be qualified for copyright protection and to make it easy for the court when determining substantial similarity by comparing the copyrighted work and the alleged infringing work. If the UK court were not to dissect the work, the consideration of a dichotomy between ideas and expression at each level of abstraction will be impossible because non-literal elements will not be apparent to the judge. As a result,

---

[44] Ibid., at 291.

the UK court will not be able to determine whether the copying in question constitutes substantial similarity.

With regard to the second issue, Ferris J's slight change of the comparison step from comparing the alleged infringing work with the core of protectable expression of the copyrighted work to comparing the alleged infringing work with the copyrighted work as a whole may yield a different result from the original three-step test. This is because, if Ferris J's approach is taken, the part substantially copied may not be the part which is copyrightable, or the part copied may be a qualitatively substantial part but, for policy reasons, it should not receive copyright protection, such as the part which generates a screen display. Therefore, the finding of infringement may be incorrect.

With regard to the third issue, Ferris J's application of the three-step test to the part which has been copied (in order to consider whether or not it represents a substantial part of the plaintiff's program) would amount to reversing the procedure of the original three-step test and would arguably represent a different concept. This thesis argues that Ferris J's approach amounts to reversing the procedure of the original three-step test because, in considering the question of substantiality, Ferris J considered the similarities between the programs first and then considered whether or not the similarities represented a substantial part. As substantiality is decided by quality, Ferris J used the three-step test to assess the quality, hence substantiality.[45] It is submitted that this is not a real application of the three-step test. It is merely the use of the filtration step which comprises three existing copyright doctrines that limit the scope of copyright protection. Moreover, the use of the three-step test to assess the quality is arguably not appropriate because the original purpose of the three-step test is to filter out the unprotectable elements and to help the court determine substantiality, not to assess the quality. More importantly, applying the three-step test to determine substantial similarity is different from applying the test to determine quality. For this reason, there is no doubt that Ferris J found difficulty in applying the abstraction step of the three-step test because, in fact, what he used is only the filtration step, not every step of the three-step test. Thus, it can be argued that Ferris

---

[45] *John Richardson v Flanders*, supra note 26, at 500.

J's version of the three-step test is confusing and will cause uncertainty when it is applied to Java programs. The inappropriateness of the modified three-step test is a strong indicator that the introduction of a new test or method of determining a copyright infringement, which is suggested by this thesis, is required in order to cope with the recent development of computer technology.

Before introducing a new method of determining an infringement, it is necessary to consider whether in fact the three-step test is reasonably based on the principle of copyright law, or it has no sound basis at all and should instantly be rejected as was in *Ibcos v Barclays*.

In contrast to Ferris J's opinion, the three-step test was rejected by Jacob J in *Ibcos v Barclays*. The importation by Ferris J of the US approach which is based on a different statute was very much the subject of criticism. In *Ibcos v Barclays*, Jacob J was of the opinion that the route of going via United State case law, that is, going via the complication of the concept of a "core of protectable expression", was not only especially unhelpful but also making the matter complicated so far as English copyright law was concerned.[46]

It can be said that the US "dissecting" approach was rejected because Jacob J disapproved both the doctrine of idea/expression dichotomy and the aphorism "there is no copyright in an idea", as they were likely to lead to confusion of thought.[47] In his view, contrary to the US approach, copyright could subsist in an idea which was sufficiently detailed. Only when the idea was sufficiently general, the taking of that idea was not regarded as copyright infringement.[48] The speech of Lord Oliver in *British Leyland v Armstrong*[49] was referred to by Jacob J as the authority for his approach that there was copyright in a detailed idea. Oliver L J expressed that it was still an infringement, although what were being copied were the engineering principles that went into the original design. Lord Oliver said:

---

[46] *Ibcos v Barclays*, supra note 27, at 302.
[47] *Ibcos v Barclays*, supra note 27, at 291.
[48] Ibid., at 291.
[49] [1986] RPC 279.

What the plaintiffs seek to protect is not the idea of an exhaust system, but their monopoly in the drawings of this particular exhaust system, the salient feature of which is the flow-line into which months of research have gone, and for the embodiment of which the drawings, taken together, form a complete set of working instructions.

The question in every case is no doubt, whether a given three-dimensional object "reproduces" the substance of the two-dimensional drawing or drawings, but I can see nothing in the Act which justifies a distinction being drawn, so far as the ambit of "reproduction" is concerned, between drawings which are purely functional, and drawings which have some aesthetic appeal, I can find no context at all for giving the word some different and more extended or restricted meaning according to the intention of the author, or the emotional response of the beholder.[50]

Following this authority, Jacob J observed that the principle "no copyright in an idea" had no relevance in the UK law once one went beyond a mere general idea; UK copyright could not prevent the copying of a mere general idea but could protect the copying of a detailed "idea".[51] It is submitted that Jacob J's approach is inconsistent with: a) previous UK authorities, b) copyright statute and c) the underlying concept based on the Software Directive. This is discussed below.

a. Previous UK authorities

It has been well established that copyright is not concerned with the reproduction of ideas, but with the reproduction of the form in which ideas are expressed.[52] 'Ideas, it has always been admitted, … are free as air'.[53] It was said by the House of Lords in the early days when the printing technology was first introduced that 'words are free to all; he may print any words that he can compose or get composed; but it does not follow that he may transcribe the composition which another has appropriated'.[54] This illustrates accurately the fundamental principle of copyright that copyright is not designed to protect ideas but only the expression of ideas. Indeed, this was made

---

[50] Ibid., at 296.
[51] *Ibcos v Barclays*, supra note 27, at 301-302.
[52] *Copinger and Skone James on Copyright*, supra note 7, at 30.
[53] A. Birrell, *The Law and History of Copyright in Books* (Cassell and Co.: London, 1899) p. 167.
[54] *Jefferys v Boosey* [1854] 4 H.L.C 815, 871 (H.L.).

clear by Lindley L.J. of the Court of Appeals in *Hollinrake v Truswell*[55] that 'copyright ... does not extend to ideas, or schemes, or systems, or methods; it is confined to their expression; and if their expression is not copied the copyright is not infringed'.[56] His Lordship went on to adopt the approach of the US Supreme Court in *Baker v Selden*, stating that this case illustrated this principle very well.[57] Therefore, it can be said that the idea/expression dichotomy doctrine and the maxim "there is no copyright in an idea" have been recognised in the United Kingdom since the 19th century. Even under the Copyright Act 1956 on which the CDPA 1988 is based, two High Court judges held that 'it stands accepted that copyright gives protection only to the form and not to the idea'[58] and 'indeed, for this proposition one need go no further than page 1 of the current edition of Copinger on Copyright'.[59] This was upheld by Russell L.J. of the Court of Appeal.[60] Indeed, the Court of Appeal confirmed this principle in its recent decision in *Designers Guild v Russell Williams.*[61] In considering the question of law, Morritt L.J. stated that:

> [it] is convenient to start from certain elementary principles. First, copyright subsists, not in ideas but in the form in which the ideas are expressed. Thus it is not an infringement of the copyright in the expression of the idea to take the idea and to apply it so long as that application does not involve copying the expression.[62]

Therefore, this thesis humbly suggests that Jacob J's rejection of the doctrine of an idea/expression dichotomy and of the aphorism "there is no copyright in an idea" is incorrect as the long standing authorities mentioned above are unevenly disregarded.

In addition, Jacob J's suggestion that – if the idea is sufficiently detailed, it can attract copyright protection – carries an unacceptable degree of uncertainty. At any rate, the concept of "detailed ideas" does not represent a clearer view than the idea/expression dichotomy. The court would not be able to determine easily which ideas should be

---

[55] *Hollinrake v Truswell* [1894] 3 Ch. 420 (C.A. Chancery Division).
[56] Ibid., at 427.
[57] Ibid.
[58] *J.C. Gleeson v H.R. Denne Ltd.* [1975] R.P.C. 471, 487 per Whitford, J.
[59] *E. Gomme Ltd. v Relaxateze Upholstery Ltd.* [1976] R.P.C. 377, 390 per Walton, J.
[60] *J.C. Gleeson v H.R. Denne Ltd.* [1975] R.P.C. 471, 488.
[61] *Designers Guild Ltd. v Russell Williams (Textiles) Ltd.*, Case No: CHANF 1998/0274/3, <http://wood.ccta.gov.uk/courtser/judgements/nsf/> accessed on 04/7/00.

considered "detailed ideas" and deserve copyright protection. When it comes to an explanation of this concept, even the latest edition of Copinger on Copyright cannot make it clear. According to Copinger on Copyright, this concept is explained as follows:

> If, however, the idea is worked out in some detail in the plaintiff's work and the defendant reproduces the expression of that idea, then there may be an infringement. In such a case, it is not the idea which has been copied but its detailed expression.[63] (underlines added)

It can be seen that, when the concept is explained, what is actually protected is expression, not the idea itself. It is submitted that although the idea is worked out in detail, it will not be protected until it has been expressed in a particular form. A real-life example of non-protectable detailed ideas is well illustrated in *Donoghue v Allied Newspapers Ltd.*[64] where the plaintiff attempted to claim copyright of his detailed ideas, (information about his racing career) which he passed to a journalist who in turn expressed and made it up into an article. Farwell J, in supporting the principle "there is no copyright in an idea", stated that:

> A person may have a brilliant idea for a story, or for a picture, or for a play, and one which appears to him to be original; but if he communicates that idea to an author or an artist or a playwright, the production which is the result of the communication of the idea to the author or the artist or the playwright is the copyright of the person who has clothed the idea in form, whether by means of a picture, a play, or a book, and the owner of the idea has no rights in that product.[65]

Therefore, it can be seen that even though the idea has been worked out in a fine detail, if it has not been recorded or expressed in some forms there will be no such thing as copyright at all. For this reason, it is submitted that in fact UK case law does support the three-step test established by the US Court of Appeals, the Second Circuit. The rejection of the three-step test by Jacob J. should not be considered as

---

[62] Ibid., at para 14.
[63] *Copinger and Skone James on Copyright*, supra note 4, at 31.
[64] *Donoghue v Allied Newspapers Ltd.* [1938] 1 Ch. 106.
[65] Ibid., at 109.

representing a good authority for UK copyright law. However, as discussed above, the three-step test is not perfectly appropriate for the latest Java computer technology. It needs to be adjusted as shown at the end of this chapter.

b. Copyright statute

It is clear that Jacob J takes an approach that there can be an infringement of copyright under the CDPA 1988 by taking the detailed idea of a work[66] and that there is no basis for the aphorism "there is no copyright in an idea" in the CDPA 1988.[67] Although the CDPA 1988 does not provide directly at the beginning of the Act that there is no copyright in an idea, it can be implied from related provisions that such aphorism does exist. Section 296A provides:

> (1) Where a person has the use of a computer program under an agreement, any term or condition in the agreement shall be void in so far as it purports to prohibit or restrict–
>
> ...
>
> (c) the use of any device or means to observe, study or test the functioning of the program in order to understand the ideas and principles which underlie any element of the program.

It can be seen that this section reinforces the idea/expression dichotomy and implicitly endorses the aphorism "there is no copyright in an idea". Therefore, it can be argued that Jacob J's observation of the CDPA 1988 is not complete and should not be regarded as representing a true situation of UK copyright law.

c. The underlying concept based on the Software Directive

As the United Kingdom has amended the CDPA 1988 to implement the Software Directive, it suffices to say that the provisions of the CDPA 1988 should be construed in accordance with the Software Directive, and that the underlying concept, if different, should have changed to conform to the that of the Software Directive for the purpose of harmonisation of the legal protection of computer programs throughout the

---

[66] *Ibcos v Barclays*, supra note 27, at 291.
[67] *Ibcos v Barclays*, supra note 27, at 289.

European Community. Based on this assumption, this thesis argues that UK copyright law should now recognise that 'only the expression of a computer program is protected and that ideas and principles which underlie any element of a program, including those which underlie its interfaces, are not protected by copyright'.[68] Thus, Jacob J's suggestion that there can be copyright in an idea which is sufficiently detailed is inconsistent with the underlying concept of both the CDPA 1988 and the Software Directive. To follow his suggestion, the United Kingdom may depart from the harmonised approach of other Member States.

From the discussion above, it can be seen that Jacob J's rejection of the idea/expression dichotomy creates an unorthodox principle in UK copyright law, that is, copyright can protect detailed ideas. This results in Jacob J's denying to follow the three-step test as he said that he did not find the route of going via United States case law particularly helpful.[69] At this point, there are some scholars who view that Jacob J did not find the three-step test helpful in this context because the *Ibcos* case involved literal copying whereas the *Altai* case concerned non-literal copying.[70] Thus, the three-step test may not apply to a literal copying case and this is the reason why Jacob J did not find the three-step test helpful.[71]

This thesis argues that in fact the three-step test can be applied equally well to a literal copying case, and that the reason Jacob J did not find it helpful is because he rejected, at the beginning, the well established doctrine of an idea/expression dichotomy on which the three-step test is based, not because the case before him was a literal copying case. A good example of the application of the three-step test to a literal copying case is found in a recent case before the US Eleventh Circuit Court of Appeals.

---

[68] Recital 13 of the Software Directive. At the international level, although the idea/expression concept is not mentioned in the Berne Convention, it is recognised by the TRIPs Agreement which provides that 'copyright protection shall extend to expressions and not to ideas, procedures, methods of operation or mathematical concepts as such'. Similarly, Article 2 of the WIPO Copyright Treaty recognises the concept in the same terms as those of the TRIPs Agreement; see J.A.L Sterling, *World Copyright Law* (Sweet & Maxwell, 1998) p. 191-192.
[69] *Ibcos v Barclays*, supra note 27, at 302.
[70] Euan Cameron, 'Lotus v Borland: An Interim for the United Kingdom' (1996) 10 International Review of Law, Computers and Technology 169, 170.
[71] *Ibcos v Barclays*, supra note 27, at 292.

In *Bateman v Mnemonics*,[72] the Eleventh Circuit applied the three-step test, stating that a parallel type of analysis must be undertaken in examining alleged instances of literal copying of computer code or screen displays.[73] The Eleventh Circuit went on to say that 'whether one chooses to call the consideration of such generally recognised challenges to literal code copying as merger and efficiency "filtration" is of little consequence; what matters is that these well-established "defences" are considered'.[74] The Eleventh Circuit also emphasised the application of the scene a` faire doctrine to the literal copying of code. The Eleventh Circuit quoted the opinion of Professor Nimmer for the proposition that 'in many instances it is virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques'.[75]

It also agreed with the opinion of the Second Circuit in *Computer Associates v Altai* that 'programmers are often constrained in their design choices by "extrinsic considerations" including 'the mechanical specifications of the computer on which a particular program is intended to run' and 'compatibility requirements of other programs with which a program is designed to operate in conjunction'.[76] The Eleventh Circuit also followed the opinion of the Ninth Circuit in *Sega v Accolade* that 'computer programs contain many logical, structural, and visual display elements that are dictated by ... external factors such as compatibility requirements and industry demands' and that 'in some circumstances, even the exact set of commands used by a programmer is deemed functional rather than creative for purposes of copyright'.[77] Consequently, the Eleventh Circuit was of the opinion that external considerations such as compatibility may negate a finding of infringement. In this case, it was clear that the jury was neither properly instructed about the connection between the three-step test and literal copying nor about the legal consequences of copying elements dictated by the compatibility requirement in this circumstance.[78]

---

[72] *Bateman v Mnemonics Inc* 79 F.3d 1532 (11th Cir. 1996).
[73] Ibid., at 1545.
[74] Ibid.
[75] Nimmer, supra note 35.
[76] Judge Birch cited the opinion of the *Altai* court (982 F.2d at 709-710).
[77] Judge Birch cited the opinion of the *Sega* court (977 F.2d at 1524).
[78] The Court of Appeal assumed that the district court failed to instruct this fact to the jury because the it was a consequence of the district court's initial failure to define the term "compatibility" and to instruct the jury on the legal consequence of copying elements dictated by compatibility requirement, together with the "Nimmer instruction" which was limited to non-literal similarity. However, Judge

Thus, the district court's reversible error was remanded by the Court of Appeals for a new trial. From this persuasive authority, it can be suggested that the three-step test can be applied comfortably to a literal copying case.

In essence, although Jacob J declined to follow the US approach, when he came to the question "was there copying from the copyrighted work?" he said:

> If the resemblance is sufficiently great then the court will draw an inference of copying. It may then be possible for the defendant to rebut the inference – to explain the similarities in some other way. For instance he may be able to show both parties derived the similar bits from some third party or material in the public domain. Or he may be able to show that the similarity arises out of a functional necessity – that anyone doing this particular job would be likely to come up with similar bits. So much is common ground between the parties.[79]

Does this statement echo the notion behind the filtration step of the three-step test? Or does Jacob J unconsciously carry out a parallel analysis by employing a different methodology? It is submitted that the answers are in the affirmative. Therefore, it can be implied that English law has more or less inherently endorsed some of the principles used by the Second Circuit in the filtration test.

However, as discussed previously, the three-step test itself is not appropriate for testing computer programs written in Java and it seems that the UK court's application of this test makes the matter more complicated and confusing. Therefore, this thesis seeks to suggest a test for substantiality for programs written in Java based on UK authority and legislation.

Birch expressed that 'it is an incorrect statement of the law that interface specifications are not copyrightable as a matter of law' but he did not explain the reason. It may be inferred that copyright can subsist in a program which has or presents interface specifications. In general, it will be an infringement of the copyright in the program if a person copies it without authorisation from the owner of the copyright. But if that person copies it in the course of creating an independent and compatible program, the copyright in the interface specification program should be denied because it is an element dictated by "extrinsic consideration". It should be noted that the Court of Appeals in this case considered that the similarity due to the compatibility requirements applied only at the literal level, not at non-literal level.

[79] *Ibcos v Barclays*, supra note 27, at 296-97.

As the nature of Java programs is object-oriented, the thesis suggests that the court can draw an analogy to the work of compilation. Among biding and persuasive authorities for determining the subsistence and infringement of copyright in compilation, the House of Lords' decision in the case of *Ladbroke (Football) Ltd. v William Hill (Football) Ltd.* is recommended as best serving as a good guide for determining "substantially similar" in Java programs. This is because a football coupon (a copyright work in question) is distinct from other compilations such as telephone directories or catalogues but has a similar nature to Java programs as skill and judgement, rather than labour, have been spent in creating the work. However, this should be combined with the consideration suggested by Ferris J in *John Richardson v Flanders*. This will be discussed below.

As the Java program has the nature of compilation, the court should consider whether the program in question, as a whole, has copyright as opposed to seeking the "core of protectable expression". Originality, which is the sole criteria for the subsistence of copyright, should be considered from the amount of skill and judgement that have been involved in creating the program. That is to say, the court should determine whether the plaintiff has spent reasonable skill and judgement in selecting and arranging the object in creating a new Java program.

However, one effect of using the *Ladbroke* approach is that unprotectable elements will be given protection and when copied from those elements it will constitute copyright infringement. In the *Ladbroke* case, the House of Lords considered the wager and the list of matches in the respondent's coupon as the element that could attract copyright for the reason of a unique selection and arrangement. If this were to apply directly to the Java program as a compilation, it may lead to an undesirable result, namely that objects which are essential for interoperability will be given copyright protection as well as its unique arrangement dictated by practice in the software industry. Therefore, the *Ladbroke* approach needs to be adjusted by incorporating the five external factors suggested by Professor Nimmer. The five external factors should be adopted and rejected as follows:

The original five factors are:

1) *the mechanical specifications of the computer on which a particular program is intended to run;*

This factor may be disregarded because a computer program written in Java can be complied to a form that can be run in any computer (platform) with the assistance of the interpreter program (Java Virtual Machine (JVM)).

2) *compatibility requirements of other programs with which a program is designed to operate in conjunction;*

With Java technology, the court can consider compatibility requirements between application programs only. The court does not need to consider the compatibility requirement between the application program and the operating system.

3) *computer manufacturers' design standards;*

This factor can also be ignored as the Java program does not need to comply with the manufacturers' design standards, unless it is compiled to a stand alone program.

4) *demands of the industry being serviced; and*

5) *widely accepted programming practices within the computer industry.*[80]

These two factors are still necessary as the demands of the industry and the practice of programming a computer program may dictated the selection and arrangement of objects.

Accordingly, it must follow that in determining copyright infringement by comparing the end products which are Java programs, the court should first consider whether the original Java program in question has copyright. This is to consider the selection and arrangement of objects that have been assembled to create the original Java program. If by the selection and arrangement of objects it can be said that copyright subsists in that program, the court will go on to consider whether the allegedly infringing Java program infringes copyright in the original program. For the infringement question, the court should compare only the selection and arrangement of objects between the original program and the alleged infringing program. The court should not consider the details of objects themselves, unless such objects are originally created by the

---

[80] Nimmer, supra note 35, at 13—66-71, cf. *Altai*, at 709.

author of the original program. If this is the case, factors number 2, 4 and 5 of Nimmer's test described above should be taken into account to limit the scope of copyright in the original Java program in question.

With regard to the method of determining "substantial similarity", it is submitted that the method refined by the Court of Appeal in *Designers Guild v Russell Williams* can be applied to software cases. In this case, Morritt L.J. defined some pointers, which may assist in answering the question of copyright infringement, as follows.[81]

1) The part to be considered is the part of the copyright work which has been copied.
2) The copying which is relevant is the copying, not of the idea, but of the expression of the idea.
3) Substantiality is a qualitative not a quantitative test.
4) The antithesis of "substantial" is "insignificant".
5) In considering whether the part which has been copied is substantial, no weight is to be attributed to that which is commonplace or well-known or derived from some other source.
6) It must be borne in mind that the object of the law of copyright is to protect the product of the skill and labour of the maker not to confer on him a monopoly in the idea it may express.

Applying these pointers to software cases, especially software which has been written in the Java language, the court has to look at the part which has been taken from the original program, not the part of the new program which can be said to copy from the original program. This is the important starting point because if the court is confused at this stage, it will render the application of the pointers wrong.

Then, the court has to bear in mind that only the expression is taken into account, not the idea even if it is a detailed idea. When judging the copying, the court needs to apply the qualitative test, not the quantitative test. This may differ from one program to another as there are many kinds of programs on the market nowadays. Some of which may possess a highly original element in every part, such as the bespoke

---

[81] *Designers Guild v Russell Williams*, supra note 61 at para 20.

program; some may not so. Thus, even if the part copied is large, if it is insignificant it cannot be considered substantial.

When considering the substantial part that has been copied, the court must not consider that some of those substantial parts are commonplace or in the public domain. This may cause little problem for programs written in the conventional way. However, for Java programs, the court has to pay special attention to this pointer. This is because Java programs normally consist of elements taken from the public domain but they are arranged in a new pattern. Therefore, the court has to consider the pattern and arrangement of the elements that have been copied, rather the elements themselves.

Finally, the court has to take into consideration that that which should be granted copyright protection is the skill and labour of the author of the program which have gone into the creation of the program, not the idea or function of the program. Therefore, if the new program, whether a conventional program or a Java program, does not appropriate unfairly the skill and labour of the author of the original program, it should not be an infringement of copyright.

6. CONCLUSION

From the discussion of the proposed framework above, it can be seen that this thesis attempts to explain the legality of reverse engineering by illustrating the actual processes of analysis and creating a new product based on the information retrieved from the reverse engineered program. Once it is clear that there are two separate processes involved in the creation of a new product – namely, the reverse engineering process and the forward engineering process – the court will not find difficulty in applying the law to control the practice of each process. It then becomes indisputable that the practice of reverse engineering should not be an infringement of copyright because the actual purpose of reverse engineering is solely for understanding the existing work. This has long been allowed in copyright law. Accordingly, it is preferable to shift the emphasis of the regulations to the practice of forward engineering because it is the stage where an infringing product would be created.

As explained above, it would be difficult in practice to prove that the forward engineering in question would lead to an infringing product. Therefore, the most appropriate solution in this circumstance is to focus on the comparison between the plaintiff's original product and the defendant's final product in order to determine infringement. This thesis' suggested test for determining infringement will provide a maximum protection to the work of the plaintiff that is created by the Java programming language. Therefore, if reverse engineering and forward engineering are to be legalised as suggested by this thesis, the software manufacturer can be assured that its computer programs are still conferred adequate copyright protection.

This thesis speculates that the proposed framework may affect some existing areas of law such as the law of confidence and patent law. This will be discussed in the next chapter.

CHAPTER FIVE

IMPACT ON THE LAW OF CONFIDENCE

1. INTRODUCTION

One branch of intellectual property law that is closely related to copyright law is the law of confidence. In many software lawsuits brought against a copyright infringer, the plaintiff may also bring a claim of breach of an obligation of confidence.[1] This is due to the fact that computer programs may be treated as confidential information which is disclosed for a limited purpose. Sometimes the plaintiff may not succeed in protecting his or her computer programs under copyright law but instead under the law of confidence. Due to the related nature of these two branches of law, it is arguable that a change proposed to the former may affect the scope of legal protection provided by the latter.

As the impact of the proposed framework on the law of confidence can reasonably be expected once the proposed framework is established, this Chapter will analyse such

an impact in order to answer the question as to what extent the scope of the law of confidence would be changed. First of all, the fact that, in the development of every piece of law, public policies have played an important role must not be ignored. This also applies to copyright law and the law of confidence. Public policies have influenced the authority to develop the law in certain ways. In order to speculate what the impact might be, it is appropriate to consider the public policies of each law in order to assess preliminarily whether the proposed framework will affect the law of confidence significantly. This will be discussed in section two. This thesis will endeavour to show that, in fact, the public policies of law of confidence do support the proposed framework, although the fundamental concept of the law of confidence may be in conflict with that of copyright law (because, firstly, the former protects ideas and information, which are confined under an obligation of confidence, but the latter does not so and, secondly, the former operates against those who receive ideas in confidence but the latter is good against the world).[2] Therefore, it is anticipated that the impact of the proposed framework on the law of confidence would not be considerable. Such analysis will be shown in section three.

Further on from section three, which mostly concerns the general impact on the law of confidence, section four will focus on a specific issue, that is, the impact of an obligation of confidence which arises in the course of employment. This is because software lawsuits often involve the unauthorised use of information contained in computer programs by the employee or the ex-employee. As opposed to section four, section five will attempt to forecast the impact of the proposed framework on the law of confidence in the circumstance where there is no obligation of confidence or where confidential information is obtained by an improper means. Finally, the analysis in this Chapter will finish with speculation of the scope of the law of confidence under the proposed framework.

---

[1] For instance, *Ibcos Computer Ltd. v Barclays Mercantile Highland Finance Ltd.* [1994] F.S.R 275; *Harbor Software, Inc. v Applied Systems, Inc.* 925 F. Supp. 1042 (S.D.N.Y., 1996); *Alcatel USA, Inc. v DGI Technologies, Inc.* Case no. 97-11339 (5th Cir., 1999).

[2] Copinger and Skone James on Copyright 13th ed., (Sweet & Maxwell: London, 1991) para. 21-2. As Lord Denning remarked in *Fraser v. Evans* [1969] Q.B. 349 at 361-2, *à propos* a written report, copyright does not subsist in the information contained in the report. It exists only in the literary form in which the information is dressed. But if the ideas or information have been acquired by a person under such circumstances that it would be a breach of good faith to publish them and he has no just cause or excuse for doing so, the court may grant an injunction against him.

2. CONFLICT AND CONSISTENCY OF PUBLIC POLICIES

As this thesis proposes to make a change in an area of the law of copyright regarding computer programs, it is stated at the beginning of this chapter that the proposed change may have an impact on the law of confidence because of their related natures. This leads to some difficulties or problems in reconciling the law of confidence with the law of copyright. It can be seen that these difficulties are derived from the conceptual difference between the law of confidence and copyright law. The conceptual difference is that the former protects ideas while the latter protects expression but not the ideas and, if the latter clearly provides that the ideas are not protected, the legal protection for the ideas provided by the former may be unenforceable. In other words, once the proposed change in copyright law is established, it would seem at first glance that the framework of the law of confidence might be distorted or, perhaps, destroyed. The reasons will be explained in this section and, later on in this section, this thesis will attempt to show that public policies may come into play to reconcile the difference between the concepts of these two pieces of law.

With regard to the concept of copyright law, it is internationally recognised that copyright law is used as a mechanism to stimulate innovations and, at the same time, to secure a fair return for the owner of copyright of the work. To achieve this goal, copyright protection is limited to the expression of ideas only, thereby leaving ideas to flow freely within the society to be used by others to develop new works.

In the US, the concept of copyright law has been made clear by the Supreme Court in *Twentieth Century Music Corp. v Atiken*[3] stating that '[t]he immediate effect of our copyright law is to secure a fair return for an author's creative labour. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good'.[4] This approach has been endorsed and followed in subsequent Supreme Court and Courts of Appeals cases.[5] Although copyright law affords protection to authors as an incentive to create new works, it must appropriately limit the extent of that

---

[3] *Twentieth Century Music Corp. v Atiken* 422 U.S. 151 (1975).
[4] Ibid., at 156.
[5] See *Sony Corp. v Universal City Studios, Inc.* 464 U.S. 417 (1984) at 430; *Computer Associates v Altai* 982 F.2d 693 (2d Cir. 1992) at 695.

protection. This is because it has to avoid the effects of monopolistic stagnation and to balance between the interests of authors in the control and exploitation of their writings and discoveries on the one hand, and society's competing interest in the free flow of ideas, information, and commerce on the other hand.[6]

In order to execute the concept of copyright law mentioned above, the limitation of the scope of the copyright holder's statutory monopoly has to be drawn so as to balance the two opposite interests. The basic limitation of copyright, which is relevant to the discussion in this section, is the separation of protected and non-protected subject matters in a work. It has been considered by the Supreme Court since *Baker v Selden*[7], which was subsequently endorsed in *Mazer v Stein*[8], that copyright protection is given only to the expression of the idea - not the idea itself.

Similarly, the World Intellectual Property Organisation (WIPO) has recently recognised the same concept of copyright law. It introduces the WIPO Copyright Treaty, which it is stated in Article 2 that '[c]opyright protection extends to expressions and not to **ideas**, procedures, methods of operation or mathematical concepts as such'.[9] As one of the signatories, the UK is bound to develop its copyright law in the same direction.

Although there is a judgment of the High Court that copyright may subsist in detailed ideas,[10] it has not been endorsed by the appellate court. In fact, the Court of Appeal and the House of Lords have consistently held that copyright law does not extend protection to ideas underlying a copyrighted work.[11] Neither of them has until now overturned such decisions. Therefore, it can be said that the position of UK copyright law is that ideas are not protected by copyright even if they have been detailed. Indeed, the line of High Court authorities commencing with *Kerrick and Company v*

---

[6] *Sony v Universal City Studios,* ibid., at 429.
[7] *Baker v Selden* 101 U.S. 99 (1879).
[8] *Mazer v Stein* 347 U.S. 201 (1954).
[9] WIPO Copyright Treaty, Geneva, December 20, 1996 <http://www.wipo.org/eng/diplconf/distrib/94dc.htm> accessed on 18/10/99.
[10] See Jacob J's judgment in *Ibcos Computers Ltd. v Barclays Mercantile Highland Finance Ltd.* [1994] F.S.R 275.
[11] See *Jefferys v Boosey* [1854] IV H.L.C. 815, 872; *Page v Wisden* (1869) 20 LT 435; *Hollinrake v Truswell* [1894] 3 Ch. 420, 427; *Joanna Christina Gleeson and Gleeson Shirt Company Ltd. v H.R. Denne Ltd.* [1975] RPC 471, 488, 490.

Lawrence and Company[12] and followed by *Donoghue v Allied Newspapers Ltd.*,[13] *E. Gomme Ltd. v Relaxateze Upholstery Ltd.*[14] and *Total Information Processing Systems Ltd v Daman Ltd.*,[15] confirms that copyright does not exist in ideas but in the expression of them.[16] In the academic field, it has been agreed that copyright does not protect ideas. It merely protects the expression of an idea.[17] Professor Lloyd explains that the main justification for refusing protection for an idea lies in the belief that ideas as such are too intangible, or too ethereal to be protected.[18]

When this concept of copyright law applies to computer programs, it means that copyright protects only the expression, but not the underlying ideas of computer programs.[19] Such a concept has been adopted in the European Community. The concept of copyright law has been clarified in the preamble of the Software Directive[20] that 'for the avoidance of doubt, it has to be made clear that only the expression of a computer program is protected and that ideas and principles which underlie any element of a program, including those which underlie its interface, are not protected by copyright under this Directive',[21] and this sentence is reproduced in Article 1(2).[22]

A difficulty, which is the issue discussed in this section, then arises in reality. The issue is that the framework or concept of copyright illustrated above may undermine the exercise of the law of confidence which can be used to protect ideas. This can be

---

[12] (1890) 25 QBD 99.
[13] [1938] Ch 106, at 109-110,
[14] [1976] R.P.C. 377, at 390.
[15] [1992] F.S.R 171, at 181.
[16] See Diane Rowland and Elizabeth Macdonald, *Information Technology Law* (Cavendish Publishing Limited: London, 1997) p. 50.
[17] David Bainbridge, *Intellectual Property 4th ed.* (Financial Times, Pitman Publishing: London, 1999) p. 30, 43-44.
[18] Ian J Lloyd, *Information Technology Law 2nd ed.* (Butterworths: London, 1997) p. 331, see also para 21.1-21.4.
[19] It may be noted that some scholars, e.g. Laddie, Prescott and Vitoria, do not agree with this proposition. However, in their analysis (The Modern Law of Copyright and Designs 2nd ed., (Butterworths: London, 1995) para. 2.73-2.75, they failed to explain why the detailed ideas which had not been recorded in any form would not receive copyright protection. According to their analysis, the detailed ideas were capable of receiving copyright protection. Moreover, they failed to recognise the "look and feel" principle which played an important role in the idea/expression dichotomy analysis.
[20] The Council Directive of 14 May 1991 on the Legal Protection of Computer Programs (91/250/EEC) OJ No L 122/42, 17.5.91.
[21] Recital 13, ibid., at OJ No L122/43.

explained as follows. As a work may attract more than one intellectual property right, e.g. copyright can coexist with a confidential obligation, a computer program may be protected by both copyright law and the law of confidence. In principle, the law of confidence is used as a mechanism to protect information which is regarded as confidential and, especially in the software industry, valuable. The whole concept of the law of confidence is to provide the owner of information a legal means to exclude certain kinds of information, including fundamental ideas or concepts, from public knowledge so as to protect his welfare arising from the secrecy of that information. Certain kinds of information that receive protection from the law of confidence may include ideas underlying a computer program.

When the law of confidence plays a role in protecting information contained in a computer program, ideas – which are meant to be free from legal protection by virtue of the proposed framework – may be protected by the law of confidence. This results in the possibility that the owner of the information may use the law of confidence to create a monopoly right over ideas and, in the case of computer programs, to restrain the free flow of ideas underlying computer programs – the situation that the law of copyright seeks to avoid. It can be seen that the concept of the law of confidence is opposite that of copyright law. Therefore, this may make it difficult to bring into force the proposed framework. It may also be said that once it is established that copyright law does not protect ideas in computer programs with the effect that anyone can perform reverse engineering without any conditions, an obligation of confidence conferred by the law of confidence may be terminated.

However, this thesis submits that a study of public policies, which are the driving force behind the development of the law of confidence, suggests that the differences in the concept or framework of these two pieces of law can be reconciled. It also suggests that the proposed framework can fit into the present framework of the law of confidence without a great impact.

---

[22] Article 1 (2): Protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.

Public policies have been used by the court to limit the scope of the law of confidence to prevent the over-exercise of this law. Generally, on the ground of public policies, a man cannot be restrained, either by contract or confidence, from using, after his employment is terminated, the personal skill, aptitude and experience acquired by him during his employment.[23] A good example stems from the Court of Appeal's opinion in *Roger Bullivant Ltd. v Ellis*[24] on the "springboard" doctrine, which prevents a person from using information obtained in confidence for activities detrimental to the person who made the confidential communication. The Court of Appeal was of the opinion that an injunction in a "springboard" case should not normally extend beyond the period for which the unfair advantage may reasonably be expected to continue. This authority was followed in *Electro Cad Australia Pty Ltd. v Mejati RCS SDN BHD*[25] where it was held that "it is obvious therefore that the effect of the "springboard" doctrine is that it is open to the Court to restrict the use of the plaintiff's information for such time as it is reasonable to expect the defendant to have used its own labour and skill to produce the same information, possibly by way of reverse engineering".[26]

Public policies may come in various forms of limiting the scope of the law of confidence. For example:

(1) The court will never uphold a covenant taken by an employer merely to protect himself from competition by a former employee;

(2) There must be some subject matter which an employer can legitimately protect by a restrictive covenant;

(3) Protection can only be legitimately claimed for identifiable objective knowledge constituting the employer's trade secrets with which the employee has become acquainted during his employment;

(4) Protection cannot be legitimately claimed in respect of the skill, experience, know-how and general knowledge acquired by an employee as part of his job

---

[23] *Copinger and Skone James on Copyright 13th ed.* (Sweet & Maxwell: London, 1991) para 21-20 p. 741 quoting *Commercial Plastics Ltd. v Vincent* [1965] 1 Q.B. 623 at 641; *Ansell Rubber Co. Pty. v Allied Rubber Industries Pty. Ltd.* [1972] R.P.C. 811; *Triplex Safety Glas Co. Ltd. v Scorah* (1938) 55 R.P.C. 21; and *Stevenson, Jordan & Harrison Ltd. v Macdonald & Evans* [1952] 1 T.L.R. 101.

[24] *Roger Bullivant Ltd. v Ellis* [1987] F.S.R. 172.

[25] *Electro Cad Australia Pty Ltd. v Mejati RCS SDN BHD* [1999] F.S.R. 291 (the High Court of Malaysia – Commercial Division, before Dato' R. Kamalanathan J.)

[26] Ibid., at 307.

during his employment, even though that will equip him as a competitor, or potential employee of a competitor;

(5) It must be possible to identify information used in the relevant business, the use and dissemination of which is likely to harm the employer, and establish that the employer has limited dissemination and has not, for instance, encouraged or permitted its widespread publication.[27]

It is obvious that in a software case concerning breach of confidence, the court would hesitate to impose upon the defendant a lengthy obligation of confidence in a restrictive covenant. The Court of Appeal in *FSS Travel & Leisure Systems Ltd. v Johnson*[28] agreed with the High Court judge's observation in the same case that the software industry was a very fast moving one and programs were updated and modified very frequently. As a result, a 12-month covenant was considered unusually and unacceptably long, given the absence of any direct evidence to demonstrate that that period of restraint was necessary.[29]

With particular relevance to reverse engineering activities, it was established by Morritt J. in *Alfa Laval v Wincanton Engineering*[30] that the purchaser of a product should not be precluded from dismantling the machine to find out how it works and telling anyone he pleases. In this case, the plaintiff's claim – that, even in the absence of any contractual term, equity would impose a duty of confidence on the purchaser restricting the information he could disclose and the person to whom he could disclose it by reference to the purpose for which he bought the machine – was rejected. This was because it was too wide and there was no authority for any such proposition.[31]

---

[27] *FSS Travel and Leisure Systems Ltd. v Johnson* [1999] F.S.R. 505, 512 per Mummery L.J.. His Lordship summarised well-settled legal propositions expounded in *Littlewoods Organisation Ltd. v Harris* [1977] 1 W.L.R. 1472; *Office Angels Limited v Rainer-Thomas* [1991] I.R.L.R. 214; *Lansing Linde Ltd. v Kerr* [1991] I.R.L.R. 80; and *Stenhouse Ltd. v Phillips* [1974] A.C. 391.
[28] *FSS Travel & Leisure Systems Ltd. v Johnson* [1999] F.S.R 505.
[29] Ibid., at 511.
[30] *Alfa Laval v Wincanton Engineering* [1990] F.S.R. 583
[31] Ibid., at 592.

The disputed standard contract in the *Alfa Laval* case contained a clause which would normally appear in any contract term,[32] including a contract for software licensing.[33] It reads:

> All drawings, designs or technical documents shall remain the property of the Company and are to be treated as confidential documents and may not without its consent be utilised by the buyer or copied, reproduced, transmitted or communicated to a third party. The buyer may not without the Company's previous written consent copy or reproduce or enable to be copied or reproduced any goods or any parts thereof.[34]

Morritt J. was of the opinion that this clause would preclude dismantling the machine to enable a third party to copy a part, but would not cover a case where the part was not copied but an otherwise unprotected concept was used. He saw no necessity for the implication of an implied term to prevent the use of such a concept, provided that the purchaser did not infringe any intellectual property right vested in others or any contractual term binding on the purchaser.[35] When this principle applies to a software licensing contract which usually contains a similar clause, it is clear that the court, by relying on public policies, would not normally impose upon a purchaser of computer programs an obligation of confidence not to perform reverse engineering to find out how it works and disclose information discovered thereof.

The limitation of the law of confidence by public policies illustrated above means that the court will not allow a new form of "industrial slavery"[36] to be created by use of actions for alleged breach of confidence. Therefore, it can be seen that public policies, which are the driving force behind the development of the law of confidence, do support the proposed framework which intends to stimulate the free flow of ideas and information.

However, the proposed framework may affect other areas of the law of confidence, which can be specifically discussed. This will be scrutinised in the next section.

---

[32] Ibid., at 591.
[33] See software licensing contracts of Microsoft and other companies.
[34] Supra note 30, at 591-592.
[35] Ibid., at 592.
[36] A term used by *Copinger and Skone James on Copyright 13th ed.*, supra note 23, at 741.

3. IMPACT ON VARIOUS AREAS OF THE LAW OF CONFIDENCE

In the previous section, the thesis discussed the impact of the proposed framework on the law of confidence in general, that is, the framework of the law of confidence as a whole. But in this section, various areas of the law of confidence which are expected to be affected will be examined. This section looks at three aspects as follows:

- Impact on jurisdictional foundation of the action for breach of confidence;
- Impact on the status of confidential information and categories of confidential information; and
- Impact on the imposition of an obligation of confidence.

3.1 Impact on jurisdictional foundation of the action for breach of confidence

Unlike copyright law, the law of confidence is entirely judge-made law which has developed through the accident of litigation.[37] This means that there is no single source of the law and the court seems free to draw on most of the available jurisdictional bases, such as contract, equity, property, trust, fiduciary relationships, unjust enrichment and tort.[38] However, it appears that equity and contract are the most important bases, so far as reverse engineering practices are concerned. Equity will give rise to an action for breach of confidence in the circumstance where the plaintiff's economic status is unfairly undermined and where there is no contract between parties. In contrast, contract will give rise to such an action regardless of any harm to the plaintiff's economic status.

3.1.1 Equity

The equity base can be relied on for an action for breach of confidence in two situations. First, where parties are in a negotiating period and no contract has been entered into. Second, where the information is acquired by a third party unlawfully, e.g., a third party illegally acquires a computer program and reverse engineers it to

---

[37] Henry Carr and Richard Arnold, *Computer Software: Legal Protection in the United Kingdom 2nd ed.* (Sweet & Maxwell: London, 1992) p. 22.

retrieve confidential information. In the light of reverse engineering activities, an impact on the equity base in the first situation is negligible because normally the parties would disclose information contained in their computer programs to the other for the purpose of their business. Therefore, there is no need to perform reverse engineering on the other's computer programs. The question of the impact on the jurisdictional foundation in this regard will not, as a result, commonly be raised. Nonetheless, in this circumstance, the party who obtains information is bound by an obligation of confidence not to use or disclose the information.

However, an impact on the equity base in the second situation can be perceived. Under the law of confidence, an action for breach of confidence can be based on equity, under the "springboard" doctrine, to prevent a third party, who obtains a computer program unlawfully, from retrieving confidential information. As the proposed framework will give a freedom to retrieve information by way of reverse engineering, the question is whether the proposed framework will render the equity base disqualified as a foundation for breach of confidence in this situation.[39]

At first glance, it may seem that the freedom of performing reverse engineering suggested by this thesis would make a claim for breach of confidence based on equity unactionable. However, the proposed framework has taken into account the economic advantage to be gained by the person who performs reverse engineering. Therefore, it suggests that a person should acquire the program lawfully in the first place before commencing any reverse engineering activity. This is to prevent him from taking a shortcut to retrieve information and from saving unfairly the time and expenses in which the owner of the reverse engineered program has invested. It can be seen that although the proposed framework does support the free flow of ideas and information, it does not encourage the public to use the new framework as a springboard to create a new program upon others' works unfairly. Based on this reasoning, it is clear that the proposed framework is consistent with the "springboard" doctrine under the law of confidence. Therefore, the impact on the equity foundation for a breach of confidence

---

[38] Henry Carr and Richard Arnold, ibid., at 23-24; Chris Reed, *Computer Law 3rd ed.* (Blackstone Press Limited: London, 1996) p. 219

[39] It may be noted that this is a separate question from the question "whether an obligation of confidence should be imposed on a party acquiring confidential information by improper means".

action is insignificant. An obligation of confidence can still be imposed on a person by basing it on equity.

3.1.2 Contract

As the thesis proposes that reverse engineering should be allowed regardless of any contractual prohibition,[40] the issues are 1) whether the proposed framework would destroy the contractual foundation of the law of confidence, and 2) if the answer is in the affirmative, whether or not this is justified.

Plainly, such a proposition will render a contractual obligation of confidence null and void. But it needs to consider first whether or not the law of confidence, in fact, intends to impose a confidential obligation to prevent one from performing reverse engineering. The discussion in this part focuses on the situation where a software company imposes a contractual obligation of confidence on its customers not to perform reverse engineering on its computer programs.

The line of authority starting from *Alfa Laval v Wincanton Engineering*[41] suggests that the law of confidence would not normally give protection other than to prevent one from taking a leap forwards by by-passing 'special labours in respect of the product in order to discover its secret'.[42] If 'taking a leap forwards by by-passing special labours' is penetrated or compromised, it seems that the law of confidence would not give any further protection. In such a circumstance, it is considered that merely lawfully obtaining a product by purchasing it from the market would suffice to show that one does not by pass special labours vested in that product. Morritt J in the *Alfa Laval* case was of the opinion that:

> No doubt the purchaser of a product will normally intend to use it for the purpose it is intended to serve. But provided that he does not infringe any intellectual property rights vested in others or any contractual term binding

---

[40] Usually, in a software-licensing contract, there is a clause which prohibits the act of reverse engineering.
[41] Supra note 30.
[42] Jacob J in *Mar UK v Teknowledge*, at para 32,quoting Francis Gurry, *Breach of Confidence* (1984).

> on him, I see no reason why he should be precluded from dismantling the machine to find out how it works and telling anyone he pleases.[43]

Thus, the unacceptable advantage of by-passing special labours can be compromised by legal purchase of a product in the open market. Indeed, the act of dismantling a product in the open market to receive information is commonly and commercially acceptable as is observed by Jacob J in *Mars UK Ltd v Teknowledge Ltd.*[44]:

> A reverse-engineer of any sort (whether one who intends just to copy or one who intends to learn how to make improvements) must start by examining the article, and if necessary, taking it apart to find out how it is made and works. This is true whether the article is mechanical or electrical. In the case of computer programs or chips with stored information the process may not involve physical de-construction: examination by electronic testing may do. But it comes to the same thing.[45]

Following his observation and the authority in the *Alfa Laval* case, Jacob J concluded that '[w]hat the owner has is the full right of ownership. With that goes an entitlement to dismantle the machine to find out how it works and tell anyone he pleases. ... There is nothing surreptitious in taking a thing apart to find out how it is made'.[46] Therefore, it is clear that although the proposed framework will destroy the contractual foundation of the law of confidence in this regard, it is justified and reasonable because it can be seen that the law of confidence, in fact, hesitates to impose an obligation of confidence to prevent lawful reverse engineering activities which are commonplace in every market.

Progressing along this line of reasoning, it can be seen that the proposed framework will fit comfortably with the existing law of confidence. Customers who wish to perform reverse engineering on computer programs will not be prohibited by copyright law or the law of confidence. They will then be able to use the knowledge

---

[43] Supra note 39 (the Alfa case), at 592.
[44] <http://wood.ccta.gov.uk/courtser/judgments.nsf/93f7e89393ff0e638025655a005738ac/3777d7666c1bcec9802567900002b5040?OpenDocument> and <http://wood.ccta.gov.uk/courtser/judgments.nsf/93f7e89393ff0e638025655a005738ac/213ed7ea762cdd86802567900002c8d54?OpenDocument> accessed on 7/10/99.
[45] Supra note 42, at para 29.
[46] Ibid.

and information gained from reverse engineering to develop interoperable or competing products as they please. Therefore, the commercial problems which occur from the inability to develop interoperable or competing products will be solved. It is also fair for the software company which holds copyright in the reverse engineered program because it will be able to recoup its investment in developing the program from the licensing of a copy of the program.

By the same token, consumers will not be prohibited by an obligation of confidence from performing reverse engineering for the purpose of maintenance. They will be able to perform every kind of maintenance (perfective, corrective, adaptive and preventive) without the fear of a legal suit based on the law of confidence or copyright law. Therefore, it can be said that the proposed framework will suitably support the current law of confidence. This will lead to a successful solution of commercial problems occurring from inability to perform perfective, corrective, adaptive and preventive maintenance as discussed in chapter two.

3.2 Impact on the status of confidential information and categories of confidential information

The next issue to be discussed here is whether the legality of reverse engineering would transform confidential information to non-confidential information, hence not protected by the law of confidence. In other words, if the law permits anyone to perform reverse engineering, the question will arise as to whether such an unrestrained access to source code would render source code not considered as confidential information and a person, who is under a contractual obligation not to disclose the information, will escape from such an obligation. This will be examined below.

On general principles, it seems clear that source code which is kept secret may be protected as confidential information.[47] Without the legality of reverse engineering, except for the interoperability purpose, it appears that the law discourages the public

---

[47] Henry Carr and Richard Arnold, *Computer Software: Legal Protection in the United Kingdom* 2nd ed. (Sweet&Maxwell: London, 1992) p. 30.

from finding out the secret of a computer program, i.e. how it works. Thus, its secret remains untouched. Once it is established that all kinds of reverse engineering activities are lawful, the public may find it encouraging to learn how computer programs function. This begs the question as to whether this legally and relatively easy access to source code would degrade the necessary quality of confidence and, as a result, any contractual obligation of confidence would be unenforceable because the subject matter lacks the quality of confidence.

This thesis submits that although this issue seems to be futuristic and one which the court may find it difficult to answer, an analogy can be drawn from previous authority to solve the issue. In *Schering Chemicals Ltd. v Falkman Ltd.*,[48] it was held that the confidential information was public in the sense that it could have been gleaned by a diligent search through scientific literature, yet the defendant, who had not done such a search, was restrained. By way of analogy, the court may apply this authority and hold that even though source code may be in the public domain (in the sense that once a program is distributed into the market anyone can legally discover it by way of reverse engineering), it remains confidential until it is reverse engineered and published, and a person who has not performed reverse engineering on the program himself would be bound by a contractual obligation not to disclose the detail of source code communicated to him by the confider. The rationale behind this principle was explained by Shaw L.J. as follows:

> **To extend the knowledge** or to revive the recollection of matters which may be detrimental or prejudicial to the interests of some person or organisation is not to be condoned because the facts are already known to some and linger in the memories of others.[49] **(emphasis added)**

Therefore, source code, which may have already been reverse engineered by someone but which has not yet published, remains confidential and is protected by the law of confidence. Moreover, as some information concerning the underlying ideas of a computer program may not be retrieved by way of reverse engineering, such as

---

[48] *Schering Chemicals Ltd. v Falkman Ltd.* [1982] Q.B. 1 (C.A.).
[49] Ibid., at 28.

"comments",[50] it can be argued that what is retrieved by way of reverse engineering is merely a pseudo version of the original source code. The information contained in the pseudo version will not be as rich as that in the original source code. Therefore, it may be argued that the original source code has not in fact been published. The result is that the original source code remains treated as confidential information and protected by the law of confidence. Thus, it can be said that there is no impact on the status of confidentiality of source code (but the law of confidence is unenforceable because of other factors discussed in section 3.1 and 3.3).

Some information, which can be very useful for rival companies, may not even be included in software, e.g. lists of customers and particular solutions to solve certain problems, hence not retrievable by way of reverse engineering. This information is perfectly protected by the law of confidence and the legality of reverse engineering proposed by this thesis will not affect its quality of confidence by any means.

3.3 Impact on the imposition of an obligation of confidence

The legality of reverse engineering may raise a question of the imposition of an obligation of confidence. As the proposed framework will bring into effect the freedom of reverse engineering activities, it is arguable that this will make it inappropriate to impose an obligation of confidence not to reveal the secret of a computer program by way of reverse engineering. This is because the subject matter lacks the quality of confidence. It may be said that the freedom of reverse engineering would make a computer program lacking such a quality.

The High Court has recently clarified this matter. The Court was of the opinion that it would hesitate to impose an obligation of confidence on the party who bought software from the open market. In *Mars UK v Teknowledge*, the defendant bought the plaintiff's machine which contained a computer program which could distinguish coins that customers inserted in a vending machine. The defendant reverse engineered such a program and learnt how to rewrite the plaintiff's program, i.e. how

---

[50] **Comment** - Text in a program that is not meant for seeing by the user but is meant for a statement so that the programmer or someone looking at the program can know what is going on,

to update the plaintiff's original program when coins had been changed. The issue in this case which is relevant to the discussion in this section is 'whether the defendant's activities by way of reverse engineering amount to a breach of confidence in law'.[51]

Jacob J. was of the opinion that the encrypted information in the plaintiff's program did not have the necessary quality of confidence because it was on the market and anyone could buy it. Moreover, he considered that the information contained therein could be accessed by anyone who had the skills to decrypt it. Thus, the buyer was free to go to a person who had the skills, or even the buyer could acquire the skills himself.[52] As a result, an obligation of confidence cannot be imposed on a person who buys software from the open market.

In this case, the source code was not merely converted to object code but also encrypted. It is submitted that the ratio decidendi of this case can apply directly to the common situation in the software industry where a rival company or a third person buys a computer program from the open market and reverse engineers it to obtain underlying ideas. Therefore, it would follow that since the purchaser of a computer program can perform reverse engineering by himself or contact a specialist to accomplish the task, the source code of a computer program distributed in the open market will not be protected by the law of confidence. It may be noted that the court hesitates to impose an obligation of confidence even where the information is relatively highly protected by using encrypting technology. Thus, a mere conversion of source code to object code without being encrypted would be more vulnerable to reverse engineering and, as a result, should not be protected by the law of confidence.

In this case, Jacob J. emphasised that he did not mean that 'were anyone to steal the information direct from the [plaintiff], thus saving themselves reverse engineering and de-encryption, [he] would not be liable for breach of confidence'.[53] This statement helps to explain the legal implications of the proposed framework, which will be considered in the next section.

<http://www.computerhope.com/jargon/jc.htm> accessed on 1/11/00.

[51] Supra note 42, at para 11.

[52] Supra note 42, at para 31.

4. LEGAL IMPLICATIONS OF THE PRESENCE OF OBLIGATION OF CONFIDENCE

4.1 Whether the proposed framework undermines an obligation of confidence in the course of employment

In this section, the discussion will be centred on the issue of whether the proposed framework will undermine an obligation of confidence in the course of employment. In general, an obligation of confidence in this circumstance is based on the contractual foundation. Therefore, it is arguable as to whether the legality of reverse engineering proposed by this thesis should prevail over a contractual obligation of confidence in this situation. The issue of the enforcement of confidential obligation in the course of employment is considered unique because of the fact that a person in such a position usually has the knowledge of the source code of software.

It should be noted that the issue discussed in this part will not concern the question of where the dividing line – between information which constitutes the employee's general stock of knowledge and information which constitutes the employer's trade secret – should be. But it will concern the issue of whether the employee will be under an obligation of confidence not to use or disclose the employer's information, i.e. underlying ideas gleaned from source code. If the proposed framework renders the employee being able to escape such an obligation, it can be said that the proposed framework has a negative impact on the existing law of confidence. On the contrary, if the proposed framework does not prevail over the presence of confidential obligations, it can be concluded that the proposed framework fits comfortably with the existing legal framework of the law of confidence.

It should also be noted that there are some differences between the employee's obligation of confidence to the employer and that of the consultant (e.g. an independent contractor) to his clients. For example, more information is likely to be held to be confidential and protectable in the case of the consultant than in the case of

---

[53] Supra note 42, at para 32.

the employee.[54] Therefore, the discussion in this section will be separated into two parts, namely an obligation of confidence owed by employees and an obligation of confidence owed by consultants. An emphasis will be placed on the first part as the outcome of the analysis of this first part is likely to apply to the second part.

4.1.1 Obligation of confidence owed by employees

According to the issue described above (viz. "whether, under the existing regime, the proposed framework will release the employee from an obligation of confidence") this thesis submits that the correct approach to solve this issue is to consider whether or not the freedom of performing reverse engineering would undermine the "springboard" doctrine. A further explanation would help in understanding this submission.

Under the "springboard" doctrine, which was established by Roxburgh J. in *Terrapin Ltd. v Builders' Supply Co. (Hayes) Ltd.*,[55] 'a person who has obtained information in confidence is not allowed to use it as a spring-board for activities detrimental to the person who made the confidential communication, and spring-board it remains even when all the features have been published or can be ascertained by actual inspection by any member of the public'.[56] Under this doctrine, it seems that the employee will be under an obligation of confidence not to use ideas gleaned from source code whereas a third party may be free to use ideas underlying source code which can be retrieved by way of reverse engineering. It can be argued that this may be considered unfair for the employee because anyone can simply access the same information by way of reverse engineering and why does only the employee have to be restrained from using the same knowledge? If this argument has merit, the employee should not be placed under such an obligation. The result will be that the freedom of reverse engineering proposed by this thesis will severely undermine the "springboard" doctrine.

---

[54] Reed (ed.), supra note 38, at p. 225.
[55] *Terrapin Ltd. v Builders' Supply Co. (Hayes) Ltd.* [1967] R.P.C. 375.
[56] Ibid., at 391.

However, this thesis suggests that the freedom of performing reverse engineering will not undermine the "springboard" doctrine and, hence, the employee can justifiably be placed under an obligation of confidence. This is because the aim of the "springboard" doctrine is to prevent an unfair start or competition. The time and trouble involved in dismantling or reverse engineering a product are considered to negate the assumption of an unfair start, as was explained by Roxburgh J.:

> I think it is broadly true to say that a member of the public to whom the confidential information had not been imparted would still have to prepare plans and specifications. He would probably have to construct a prototype, and he would certainly have to conduct tests. Therefore, the possessor of the confidential information still has a long start over any member of the public.[57]

Therefore, if the possessor of the confidential information is the employee, he 'must be placed under a special disability in the field of competition in order to ensure that he does not get an unfair start'[58] over a member of the public who has to take the trouble to perform reverse engineering to obtain the same information. This seems to be fair competition because usually the employee is in the position that he can obtain the information without having to expend time, effort, labour and capital. It would be very unfair in terms of economic advantage to place the employee in the same legal position as a member of the public who has to spend time, effort, etc. to obtain the same information. Moreover, the employee is in a position in which he can obtain a greater amount of confidential information than a member of the public, such as information which is not included in the program source code or which cannot be retrieved by way of reverse engineering. Therefore, it seems reasonable to retain an obligation of confidence upon the employee.

Although an argument may arise as to the fact that reverse engineering activities performed by a member of the public can also give rise to the "springboard" type of action for breach of confidence, such an argument is refuted by an observation of Roxburgh J. He observed that the possessor of the confidential information still had a good start over any member of the public because the person performing reverse

---

[57] Ibid., at 392.

engineering had to go through several steps before getting the information and using it to produce a rival product. His observation was considered and followed by Jacob J. in the *Mars* case. Jacob J. stated that only if the information was illegitimately taken, would it give rise to the action for breach of confidence under the "springboard" doctrine. This is because if a member of the public has acquired the product legitimately, he will not be considered as attempting to by-pass 'special labours in respect of the product in order to discover its secret'. Thus, it is submitted that the freedom of reverse engineering should not be seen as a way of getting information and using it as a "springboard" for activities detrimental to the confider, as opposed to the employee who is imparted confidential information and uses it as a "springboard". This kind of employee should be put under an obligation of confidence because he has a good chance to by-pass the special labours invested in the product. Therefore, it can be seen that the proposed framework does not undermine the "springboard" doctrine and can co-exist with the present regime of the law of confidence.

4.1.2 obligation of confidence owed by consultants

In general, the consultant holds an obligation of confidence towards the client in much the same way as the employee does towards the employer. However, the courts seem to have taken a rather hard line with consultants who have acquired confidential information whilst working for one client then subsequently used that information for their own benefit or for the benefit of others.[59] The issue, thus, emerges as to whether the freedom of performing reverse engineering would compromise the hard line taken by the courts.

In *Schering Chemicals Ltd. v Falkman Ltd.*,[60] the Court of Appeal held that even though the information was already known to some and lingered in the memories of others, it still was of a confidential nature and was protected by the law of confidence if the use or disclosure of the information would be detrimental to the interests of some persons or organisations.[61] Thus, the unauthorised use of the information by the

---

[58] Ibid., at 392.
[59] Reed, supra note 38 at p. 226.
[60] Supra note 48.
[61] Ibid., at p. 28.

defendant consultant was a breach of confidence.[62] Applying this authority to the issue discussed, it is highly likely that, although information contained in a computer program can be retrieved by a member of the public, the consultant is still under an obligation of confidence not to use or disclose such information. Accordingly, the freedom of performing reverse engineering proposed by this thesis does not affect the presence of an obligation of confidence owed by consultants.

The justification for this proposition seems to be based on fiduciary duty, morality and public interest. This is because the consultant is in the position where he can easily exploit the information imparted to him as a springboard for creating a new product or he can deprive his client of the lead time in the market. Hence, a strict obligation of confidence is required.

5. LEGAL IMPLICATIONS OF THE ABSENCE OF OBLIGATION OF CONFIDENCE

5.1 Whether the proposed framework approves the acquisition of confidential information by improper means

In the previous section, the legal implication of the proposed framework is analysed in the situation where an obligation of confidence commonly exists and where information is easily and frequently accessed. However, this is not the only means whereby information can be accessed. As is illustrated in Chapter two, information contained in a computer program may be accessed by hackers or spies, or can be retrieved by any other improper means for which it is not possible to give an entire list here.[63]

From a legal standpoint, there seems to be a surprising gap in the English law of confidence in protecting information acquired by improper means. This is because a

---

[62] Ibid., at p. 29.

[63] As is noted by the Court in *E.I. duPont deNemours & Co. v Christopher* 431 F.2d 1012 (5th Cir, 1970), a complete catalogue of improper means is not possible. In general, they are means which fall below the generally accepted standards of commercial morality and reasonable conduct.

person such as a hacker or spy is under no pre-existing obligation to respect confidentiality, hence no breach and no action.[64] The hacker cannot be said to have voluntarily undertaken an obligation to respect the confidentiality of the information he has improperly acquired. Any obligation must be imposed involuntarily by the law but here the law seems remarkably reluctant to intervene.[65] Although one leading scholar in this field and the Law Commission suggested a long time ago that the action for breach of confidence should be capable of extension to protect confidential information obtained by improper means regardless of whether there was a pre-existing relationship of confidence,[66] this deficiency in the English law of confidence remains.

It is worth making a brief reference to the report of the Law Commission since the establishment of the proposed framework may resurrect the consideration of the report. In the Law Commission's report, it was suggested that an obligation of confidence should arise where the person from whom the information has been obtained without his authority could reasonably expect that the information would not be so obtained, and where the acquirer of the information knows or ought to know that in receiving it he is defeating the reasonable expectations of the original holder.[67] Based on this suggestion, the Law Commission recommended that a number of situations should be specified in order to cover the most important circumstances in which the acquirer of information should be treated as subject to an obligation of confidence, e.g. in cases of industrial espionage.[68]

Under the proposed framework which supports the freedom of performing reverse engineering, an issue arises as to whether the proposed framework will have legal implications on the deficiency mentioned above in the sense that – whether it will

---

[64] Reed, supra note 38 at p. 230.
[65] Ibid.
[66] Professor Gareth Jones, 'Restitution of benefits obtained in breach of another's confidence' (1970) 86 L.Q.R. 463, at p. 482-83; The Law Commission (Law Com. No. 110), *Breach of Confidence: Report on a Reference under Section 3(1)(e) of the Law Commissions Act 1965, presented to Parliament by the Lord High Chancellor, by Command of Her Majesty October 1981* (HMSO: London, 1981). The Law Commission was requested by the Lord High Chancellor to consider, *inter alia*, what remedies should be provided for persons who suffered loss or damage in consequence of the disclosure or use of information unlawfully obtained and in what circumstances such remedies should be available.
[67] The Law Commission's report, ibid. at 114.
[68] The Law Commission's report, ibid., at 114-123, 194-195.

bring about an effect that the acquisition of confidential information by improper means is approved.

The proposed framework offers a new approach to stimulate the development of technology. Although it takes quite a radical approach, it is still based on the balance of economic interests of the parties concerned. Thus, the proposed framework suggests that the person who wishes to perform reverse engineering must acquire a product legally. In other words, he must not by-pass the special labour and investment vested in the product reverse engineered.

It can also be inferred from the proposed framework that it does support Professor Jones and the Law Commission's propositions to extend the law of confidence to cover the situation where the confidential information is acquired by improper means.[69] For this reason, it can be seen that the proposed framework does not approve the acquisition of confidential information by improper means. Nor is the act of reverse engineering done by hackers approved by the proposed framework, unless the hackers have acquired a computer program lawfully. If the hackers lawfully acquire a computer program as described, the hackers can be considered as a mere unknown lawful user who has the full right of ownership.

However, under the current law of confidence where hackers' reverse engineering activities are arguably not a breach of an obligation of confidence, it cannot be concluded that the freedom to perform reverse engineering amounts to the legal approval of any kind of hackers' activities. This is due to the fact that the proposed framework still supports the "springboard" doctrine, not to allow a third party to by-pass special labour to gain advantage of the lead-time enjoyed by the owner of information.[70]

But it is not wrong, at present, to state that the freedom of performing reverse engineering does not change the legal status of hackers' activities or the legal status of the acquisition of confidential information by improper means. Nevertheless, it is submitted that the extension of the scope of the law of confidence to prevent the

---

[69]Jones, supra note 66 at 485.

acquisition of confidential information by improper means is needed.[71] However, a discussion as to how to extend the scope of the law of confidence is beyond the scope of this thesis. It would be interesting to see such a development in this area in the future. But if the scope of the law of confidence were to be so extended, the proposed framework would have helped to prevent such an unlawful acquisition by putting up a copyright barrier against the hacker who has not acquired a computer program lawfully.

6. CONCLUSION

From the discussion above, it can be seen that the proposed framework will not greatly affect the scope of the existing framework of the law of confidence. The proposed framework enables reverse engineering to be carried out legally where the computer program is purchased (licensed) in the open market. The public policy of the law of confidence shows that it is willing to permit the dismantling of a product so that an unprotected concept can be studied. Only when the dismantling is done with the aim of copying a part will the law of confidence confer protection against such an activity. This is consistent with the proposed framework because reverse engineering only possesses the aim of learning how the product functions. Any activity beyond this point will fall within the forward engineering stage which may be prohibited by the law of confidence and the law of copyright if it has the aim of copying a part.

It can also be said that the proposed framework's suggestion that reverse engineering can be performed on a purchased product is consistent with the general principle of the law of confidence. This is because both of them prohibit the unfair taking of a leap forwards by by-passing special labours vested in the product. The purchase of the product will negate the assumption that special labours vested in the product have been by-passed. Although this may nullify any contractual prohibition which could give rise to an action for breach of confidence, it cannot be said that the foundation of the law of confidence is destroyed as explained above. On the other hand, if the

---

[70] See discussion in section 3.1.1 of this chapter.
[71] It may be noted that although the law of confidence in the UK does not guard against the acquisition of confidential information by improper means, such an acquisition may be outlawed under unfair competition law in the US.

reverse engineer does not purchase the program, nor acquire it by proper means, it is submitted that he should be bound by an obligation of confidence and he should not be able to perform reverse engineering under the proposed framework.

In the course of employment, the proposed framework will not interfere with the existing confidential relationship either between the employer and the employee or between the employer and the consultant. The employee and the consultant remain under an obligation of confidence. Therefore, they cannot use the knowledge gained during the course of employment as a springboard to the detriment of the employer by asserting that the information can readily be accessed by others because of the proposed framework.

To conclude, this thesis suggests that reverse engineering should be permissible on a legally purchased program under the proposed framework and it should be a breach of the law of confidence only when the reverse engineer by-passes the special labour vested in the program by not paying for the cost of the program. The law of confidence should also remain in force, once the proposed framework is established, in order to protect information, confined to the employee or the consultant in the course of employment, against unfair use. As the law of confidence has not been extended to cover the acquisition of the information by improper means, the thesis suggests that this should be done so that the economic interests of the owner of information will be better protected. The proposed framework in essence will support such an extension of the scope of the law of confidence.

CHAPTER SIX

IMPLICATION OF THE PROPOSED FRAMEWORK ON PATENT LAW

1. INTRODUCTION

In the previous chapter, this thesis focused on the issues of an impact of the proposed framework on the law of confidence. The discussion in that chapter was based mainly on common law. In contrast, this chapter will analyse an impact on patent law which has a statutory basis.[1] An analysis of the impact on patent law can be said to be of importance because it will help justify the proposition of this thesis.

The first issue to be solved is whether or not the proposed framework will actually have an impact on the boundary of patent law. This requires a general understanding of the current practice of patenting software and a detailed understanding of the legal

[1] The Patents Act 1977 and the European Patent Convention 1973.

position of software patentability under UK patent law. Such understandings will help establish a clear boundary of patent protection for software. Once the boundary of patent protection is made clear, it is quite straightforward to explain an implication of the proposed framework, which may be in the form of any movement in the software industry. According to this plan for solving the first issue, the next section will be an overview of the current practice of patenting software. Then, in section three, there will be an analysis of the current position of software patentability under UK patent law. The impact of the proposed framework on the current position of UK patent law will be presented in section four. It is submitted that the introduction of the proposed framework may lead to a move towards the broadening of the current scope of UK patent law and, perhaps, towards the introduction of the utility model.

The second issue is whether the impact will be a negative or positive one, i.e. whether it will obstruct or support the development of UK patent law. As UK patent law is based on the European Patent Convention 1973 (EPC), the development of UK patent law is tied up with the European Patent Office (EPO)'s jurisprudence. This renders it necessary to examine the position of software patentability under the EPC and then compare it to that under UK patent law. The result of the comparison will reveal the potential development of UK patent law. Thus, it will be seen if the impact of the proposed framework will support the development of patent law, hence justifying the proposition of this thesis. The analysis of software patentability under the EPC and comparison to UK position will be discussed in section five. Section six will illustrate the development of UK patent law. It will be shown that the impact of the proposed framework does support such a development.

The development of UK patent law is not only influenced by the EPO's jurisprudence but also by the proposed changes to the provisions of the EPC and by international patent law, although indirectly. These are the proposal for the revision of the EPC, the Community Patent, the protection of inventions by the utility model and the WIPO Patent Law Treaty. They will be observed in section seven. As will be seen, the proposal for the revision of the EPC regarding computer programs is currently a public debate in the EU, the result of which will have an effect on patent law of the member states of the EPC. The proposal suggests that the computer programs exception should be deleted from Article 52 EPC. This is the move towards the

broadest scope of software patentability, i.e. to allow computer programs as such to be patented as in the US. This raises the third issue as to whether or not the proposal should be adopted. It is submitted that the proposal should not be adopted. This chapter's answer to this issue will make a contribution to the knowledge and show how the proposed framework supports that answer. This will be discussed in section eight.

2. OVERVIEW OF THE CURRENT PRACTICE OF PATENTING SOFTWARE

Patent law can be said to serve a similar aim to copyright law, i.e. to stimulate the creation of useful art and technology,[2] but it was originally designed to protect different kinds of works from those protected under copyright law. While patent law protects articles of manufacture, i.e. something which is capable of industrial application, copyright law protects the works of aesthetic creations, e.g. literary work and artistic work, although some of these copyrighted works can also be considered functional such as engineering drawings and computer programs. This dividing line is drawn clearly in current patent law.[3]

However, the dividing line is becoming blurred by the hybrid nature of computer programs. Although computer programs have internationally been recognised as a kind of literary work under copyright law[4] and therefore have theoretically been denied a patent protection under patent law,[5] there have been successful attempts to obtain a patent for computer programs in the guise of software related inventions (SRI). For example, a European patent number EP643851B1 (a debugger program which includes correlation of computer program source code with optimised object code), number EP000767419B1 (a method and system in a data processing system windowing environment for displaying previously obscured information).

---

[2] W. R. Cornish, *Intellectual Property 4th ed.* (Sweet & Maxwell: London, 1999) p. 110-115.
[3] Section 1(2)(b) of the Patents Act 1977.
[4] Section 3(1)(b) of the Copyright Designs and Patents Act 1988, Article 1 of the Directive of 14 May 1991 on the legal protection of computer programs.
[5] Section 1(2)(c) of the Patents Act 1977, Article 52(2)(b) of the EPC.

In the United States, one further step has already been taken. Computer programs as such are now patentable. For instance, a US patent number US5920316 (taskbar with start menu). Indeed, the range of SRI now covered by patents is very broad. This includes medical diagnostic systems, mechanical fault diagnosis, neural networks, business and production management systems, CAD/CAM, process control, integrated circuit design, etc.[6] The latest trend in the application for SRI seems to be within the e-commerce arena as a result of the advent of the Internet. The United States has now taken the lead in allowing SRI for e-commerce. Examples of SRI relating to e-commerce between the years 1998 and 2000 are as follows.

US5710887: Computer system and method for electronic commerce (Broadvision Inc., 1998)[7]

US5715314: Network sales system (Open Market Inc., 1998)[8]

US5729594: On-line secured financial transaction system through electronic media (Edwin E Klingman, 1998)[9]

US5754772: Transaction service independent HTTP server-to-transaction gateway (Unisys Corp., 1998)[10]

US5768385: Untraceable electronic cash (Microsoft Corp., 1998)[11]

US5793966: Computer system and computer-implemented process for creation and maintenance of online services (Vermeer Technologies Inc., 1998)[12]

US5797127: Method, apparatus, and program for pricing, selling, and exercising options to purchase airline tickets (Walker Asset Management Ltd. Partnership, 1998)[13]

US5809144: Method and apparatus for purchasing and delivering digital goods over a network (Carnegie Mellon University, 1998)[14]

US5870562: Universal domain routing and publication control system (PFN Inc., 1998)[15]

---

[6] Michael Evans, <http://www.hobsonaudley.co.uk/cf14/11_ITLaw2.cfm> accessed on 4/11/00.
[7] http://www.delphion.com/cgi-bin/viewpat.cmd/US05710887__
[8] http://www.delphion.com/cgi-bin/viewpat.cmd/US05715314__
[9] http://www.delphion.com/cgi-bin/viewpat.cmd/US05729594__
[10] http://www.delphion.com/cgi-bin/viewpat.cmd/US05754772__
[11] http://www.delphion.com/cgi-bin/viewpat.cmd/US05768385__
[12] http://www.delphion.com/cgi-bin/viewpat.cmd/US05793966__
[13] http://www.delphion.com/cgi-bin/viewpat.cmd/US05797127__
[14] http://www.delphion.com/cgi-bin/viewpat.cmd/US05809144__
[15] http://www.delphion.com/cgi-bin/viewpat.cmd/US05870562__

US5870717: System for ordering items over computer network using an electronic catalogue (IBM Corp., 1999)[16]

US5963924: System, method and article of manufacture for the use of payment instrument holders and payment instruments in network electronic commerce (VeriFone Inc., 1999)[17]

US5987140: System, method and article of manufacture for secure network electronic payment and credit collection (VeriFone Inc., 1999)[18]

US6016484: System, method and article of manufacture for network electronic payment instrument and certification of payment and credit collection utilising a payment (VeriFone Inc., 2000)[19]

US6026379: System, method and article of manufacture for managing transactions in a high availability system (VeriFone Inc., 2000)[20]

US6026430: Dynamic client registry apparatus and method (Ronald A. Butman and others, 2000)[21]

US6061726: Dynamic rights assignment apparatus and method using network directory services (Novell Inc., 2000)[22]

US6101485: Electronic solicitations for Internet commerce (IBM Corp., 2000)[23]

US6119105: System, method and article of manufacture for initiation of software distribution from a point of certificate creation utilising an extensible, flexible architecture (VeriFone Inc., 2000)[24]

US6138119: Techniques for defining, using and manipulating rights management data structures (InterTrust Technologies Corp. 2000)[25]

From the examples above, it can be seen that the majority of patents are concerned with methods of purchasing goods over the Internet with the enhancement of security of information sent and received in such an electronic commercial transaction. The minority of those patents are concerned with the efficiency of transferring data from

---

[16] http://www.delphion.com/cgi-bin/viewpat.cmd/US05870717__
[17] http://www.delphion.com/cgi-bin/viewpat.cmd/US05963924__
[18] http://www.delphion.com/cgi-bin/viewpat.cmd/US05987140__
[19] http://www.delphion.com/cgi-bin/viewpat.cmd/US06016484__
[20] http://www.delphion.com/cgi-bin/viewpat.cmd/US06026379__
[21] http://www.delphion.com/cgi-bin/viewpat.cmd/US06026430__
[22] http://www.delphion.com/cgi-bin/viewpat.cmd/US06061726__
[23] http://www.delphion.com/cgi-bin/viewpat.cmd/US06101485__
[24] http://www.delphion.com/cgi-bin/viewpat.cmd/US06119105__
[25] http://www.delphion.com/cgi-bin/viewpat.cmd/US06138119__

one network to another or others. While both the majority and the minority have computer programs embedded in the patented inventions, the former can also be considered as a method for doing business. This confirms that US patent law not only permits SRI but also accepts claims for SRI which equate to a method for doing business.

It may be noted that it is not possible to identify what kinds of those SRI are subject to reverse engineering. This is because all reverse engineering is done in secret as there is a fear of a legal suit. Therefore, the discussion in this chapter will not attempt to answer the question of what kinds of SRI are subject to reverse engineering but rather attempt to analyse the development of patent law regarding computer programs in general to indicate an impact of the proposed framework on patent law and to answer what would be the desirable approach for the EU and the UK in the future.

3. CURRENT POSITION OF SOFTWARE PATENTABILITY UNDER UK PATENT LAW

This section seeks to illustrate the extent to which computer programs are allowed to be patented in the UK. The analysis of case law will show how the court interprets the provision of the Patents Act 1977 and how other excluded matters are applied to limit the scope of the patentability of computer programs. This will be used as a basis for the discussion on the impact of the proposed framework on UK patent law in the next section.

3.1 Computer programs not patentable as such

Under section 1(2)(c) of the Patents Act 1977, computer programs are declared as not being inventions.[26] However, they are so considered only to the extent that a patent

---

[26] Section 1(2) of the Patents Act 1977:

> It is hereby declared that the following (among other things) are not inventions for the purposes of this Act, that is to say, anything which consists of –
>
> (a) ...
>
> (b) ...

relates to computer programs as such.[27] This can be seen as the interplay between the exclusion and the exception to the exclusion which becomes the issue for discussion in this section. Therefore, the questions are, how does the court draw the line between these two and how does the court define the ambit of SRI ?

Aldous L.J. reiterates the principle of UK patent law which can be said to be the starting point of this section analysis.

> ... it is and always has been a principle of patent law that mere discoveries or ideas are not patentable, but those discoveries and ideas which have a technical aspect or make a technical contribution are. Thus the concept that what is needed to make an excluded thing patentable is a technical contribution is not surprising.[28]

From the passage above, it can be inferred that, to be patentable, computer programs have to be presented as a part of an inventive machinery or industrial process. This means that an application for a patent has to be directed to some innovative practical or technical feature that is produced by computer programs.[29] A question arises as to what kind of a technical feature or technical effect that the UK court allows to be patented. One of the leading cases in this area is *Re Merrill Lynch, Pierce Fenner & Smith Inc's Application*.[30]

In *Merrill Lynch's Application*,[31] agreeing with the Principal Examiner, Falconer J. stated that the Examiner was right in separating excluded matters from the alleged

---

(c) a scheme, rule or method for performing a mental act, playing a game or doing business, or **a program for a computer;**

[27] Section 1(2) last paragraph:
but the foregoing provision shall prevent anything from being treated as an invention for the purposes of this Act only to the extent that a patent or application for a patent relates to that thing **as such**.

[28] *Fujitsu Limited's Application* [1997] R.P.C. 608, 614.

[29] David Bainbridge, *Intellectual Property 4th ed.* (Financial Times, Pitman Publishing: London, 1999) p. 370.

[30] *Re Merrill Lynch, Pierce Fenner & Smith Inc's Application* [1988] R.P.C. 1.

[31] The claimed invention was an improved data processing system for implementing an automated trading market in securities. More specifically, it was the object of the invention to provide an automated market making system for qualifying and executing orders for securities transactions. This system comprised a computer program which controlled the working of the computer in the following manner. The system retrieved and stored the best current bid and asked prices; qualified customers' buy/sell orders for execution; executed the orders; and reported the trade particulars to customers and to the national stock price reporting system.

invention and then asking whether or not the non-excluded features were already known and obvious.[32] The most important aspect of his judgment was the way in which he interpreted the last paragraph of section 1(2). He gave his opinion as follows:

> Plainly, if a patent or patent application relates only to one of the matters specified in (a), (b), (c), or (d) of the subsection, e.g. a computer program, that would be excluded from patentability by the subsection, but in my view the excluding effect of the subsection is wider than that. It seems to me that the words "to the extent that" contemplate that the subsection is also to be applicable to cases where the invention involves one of the excluding matters (specified in paragraphs (a), (b), (c) and (d)) but does not relate to it only.[33]

He explained further that, in construing the words "to the extent that" in this way, he meant that 'there cannot be a patentable invention in so far as the invention **resides in** the computer program itself'.[34] The outcome of Falconer J.'s construction of the qualification would be that many alleged software-related inventions would be excluded from patentability. This is because a computer (a claimed invention), which is hardware, normally operates in a conventional manner when carrying out various steps of the program. Therefore, under Falconer J.'s construction, it does not matter how inventively a computer is programmed. That computer (the claimed invention) would not be patentable.

However, the Court of Appeal in *Genentech Inc.'s Patent*[35] disagreed with the approaches of Falconer J.. Purchas L.J. (the Court of Appeal) felt that Falconer J. placed an undue emphasis upon the words "to the extent that". This would effectively distort the general meaning of the provision.[36] Agreeing with Purchas L.J., Dillon L.J. pointed out that the broad interpretation of the excluded matters should not be established as a general principle because it would produce an absurd outcome when applied to other excluded matters. For example, the patenting of new drugs and medicine or microbiological processes would be denied because the application of the

---

[32] Ibid., at 12.
[33] Ibid., at 11.
[34] Ibid., at 12.
[35] *Genentech Inc.'s Patent* [1989] R.P.C. 147 (C.A.).
[36] Ibid., at 207.

discovery cannot be separated from the discovery of the new drugs or medicinal processes which are themselves novel and not obvious.[37] In fact, the correct principle is that 'while one cannot patent a discovery, if on the basis of that discovery one can tell people how it can be usefully employed, then a patentable invention may result, even though the mode of use is obvious enough'.[38] To prevent such a distorted outcome, Purchas L.J. was of the opinion that the plain and ordinary interpretation should be given to the phrases "only to the extent that" and "relates to that thing as such". This is to take the two phrases together as meaning that any of the matters listed in sub-paragraphs (a) to (d) shall not be an invention for the purposes of the Act and shall only to the extent that the application or patent relates to that step as such be disqualified.[39] As a result of this construction, those which **involve** excluded matters are still patentable. Thus, modes of using the excluded matters either in a process or in relation to an artefact are patentable. However, while Dillon L.J. rejected Falconer J.'s principle, his Lordship came to a somewhat contradictory conclusion. He said:

> it does not in the least follow that I disagree with the result of that case. It would be nonsense for the Act to forbid the patenting of a computer program, and yet permit the patenting of a floppy disc containing a computer program, or an ordinary computer when programmed with the program; it can well be said, as it seems to me, that a patent for a computer when programmed or for the disc containing the program is no more than a patent for the program as such.[40]

This passage in essence implies that inventions are not patentable if the technical effect they produce is not separable from the computer program embedded in the inventions. Nevertheless, Dillon L.J.'s reasoning, which was not consistent with his conclusion, left some doubt as to the true position of software patentability under UK patent law. If this were to be taken into account alone, it would have been difficult to ascertain the position of UK software patentability. This would in turn result in the uncertainty of the evaluation of the impact of the proposed framework on UK patent law. However, this is not the case because the examination of subsequent case law

---

[37] Ibid., at 239.
[38] *Genentech Inc.'s Patent* [1987] R.P.C. 553, 566, per Whitford J.
[39] *Genentech* (CA), supra note 35, at 207.
[40] Ibid., at 240.

shows that the UK court is reluctant to allow claims for an invention which in effect resides in a computer program.

This can be seen from *Gale's Application*.[41] The Court of Appeal had to decide whether or not a claim for a ROM (Read Only Memory) programmed to carry out a method of calculating square roots was patentable. Nicholls L.J. rejected the claim by reasoning that:

> The attraction of Mr. Gale's case lies in the simple approach that ... he has found an improved means of carrying out an everyday function of computers. ... A computer ... will be a better computer when programmed with Mr Gale's instructions. ... But the instructions do not embody a technical process which exists **outside** the computer. Nor, as I understand the case as presented to us, do the instructions solve a "technical" problem lying within the computer ... .[42] **(emphasis added)**

This passage indicates that the Court of Appeal appears to revert to the method of determining a patentable matter created by Falconer J. (there cannot be a patentable invention in so far as the invention resides in the computer program) which was once rejected by the Court of Appeal in *Genentech*.

The trend of rejecting the claim for an invention which resides in a computer program was seen again in *Fujitsu Limited's Application*.[43] This case concerned a method and apparatus for modelling a synthetic crystal structure for designing inorganic materials. In this case, the Court of Appeal (Aldous L.J.) had to consider whether or not a computer which was programmed in such a way, that the operator could select an atom, a lattice vector and a crystal face in each of two crystal structures displayed on the display unit, was patentable.[44] As the whole operation revolved around the computer program, Aldous L.J. asked if there was a technical contribution so that it

---

[41] *Gale's Application* [1991] R.P.C. 305.
[42] Ibid., at 327-328.
[43] *Fujitsu*, supra note 28.
[44] The computer in question operated in the following manner. Upon instructions, it converted data representing the physical layout of the structure that was obtained by combining the original two structures in such a way that the selected atoms, the selected lattice vectors and the selected faces are superposed. The resulting data were then displayed to give a picture of the resulting combined structure, ibid., at 618.

could be said that the invention did not consist of a computer program as such.[45] His Lordship responded to this question as follows:

> Mr. Birss [appeared on behalf of the applicant Fujitsu] is right that a computer set up according to the teaching in the patent application provides a new "tool" for modelling crystal structure combinations which avoids labour and error. **But those are just the sort of advantages that are obtained by the use of a computer program.** Thus the fact that the patent application provides a new tool does not solve the question of whether the application consists of a program for a computer as such or whether it is a program for a computer with a technical contribution. I believe that the application is for a computer program as such.[46] **(emphasis added)**

The difficulty that Aldous L.J. was facing was that what could be considered as a technical effect was residing in a computer program itself. The technical contribution, if any, in this case could be an innovative way of producing a picture of the combined structure on a screen display. However, he viewed that it was no more than the use of a computer program and the claim to that effect was no different from a claim to a computer program which was statutorily excluded. Therefore, the appeal was dismissed. It can be seen that, under UK case law, if the claimed invention whose technical effect is inseparable from the computer program itself, the claimed invention would be denied a patent.

From the analysis of the cases mentioned above, it can be concluded the position of UK patent law is such that if the technical effect or contribution resides in a computer program, it is highly likely that an application for a patent for such technical effect would be denied. This is because it is excluded by section 1(2)(c) as being a program for a computer as such. Hence, it follows that the impact of the proposed framework on UK patent law will have to be evaluated on the basis that the UK court would grant a patent only to the application which is directed to technical effect that does not reside in computer programs. There are also further limitations which make the scope of software patentability even narrower and which need to be taken into account when evaluating the impact. This will be examined below.

---

[45] Ibid.
[46] Ibid.

3.2 Application of other excluded matters to prohibit patenting of computer program

The limitation of software patentability is not only limited to computer programs as such and the technical effect which resides in computer programs as discussed above, but also to the extent that the claimed software-related invention falls within other excluded matters. The discussion in this section will show what these are.

An approach taken by the Court of Appeal (Fox L.J.) in the *Merrill Lynch* case[47] illustrates this further limitation. Fox L.J. considered that the claim in *Merrill Lynch's Application* was merely a method of doing business because a data-processing system simply made a trading market in securities.[48] Therefore, it was caught by s. 1(2)(c) of the Patents Act 1977. In his opinion, it did not matter whether or not the method of doing business might be an improvement on previous methods of doing business.[49] The prohibition in section 1(2)(c) is generic; qualitative considerations do not enter into the matter. He explained that:

> If what is produced in the end is itself an item excluded from patentability by section 1(2), the matter can go no further. Claim 1, after all, is directed to "a data processing system for making a trading market". That is simply a method of doing business. A data processing system operating to produce a novel technical result would normally be patentable. But it cannot, it seems to me, be patentable if the result itself is a prohibited item under section 1(2). In the present case it is such a prohibited item.[50]

The application of "method of doing business" will establish an authority for the courts in subsequent cases. Such an application will dramatically reduce the patentability of SRI because many computer programs in the market are designed to assist the efficiency of doing business, especially those in e-commerce.

In addition to the method of doing business, an exclusion of patentability in the form of being a "mental act" has been used recently to reject the SRI claim. In *Fujitsu*

---

[47] *Merrill Lynch's Application* [1989] R.P.C. 561 (C.A.).
[48] *Fujitsu*, supra note 28, at 569.
[49] Ibid.
[50] Ibid.

Ltd.'s Application,[51] the Court of Appeal (Aldous L.J.) upheld the decisions of the Patent Office and the High Court in rejecting the application. The Court of Appeal considered that the application was for a computer program as such or that the technical effect produced by the computer program was merely a method of performing a mental act. In this case, the Court of Appeal's decision suggested a broad interpretation of the excluded matters provided in s. 1(2)(c).

Aldous L.J. referred to his judgment in *Wang Laboratories Inc.'s Application*[52] where he was of the opinion that:

> just as a claim to a disc containing a program could be in fact a claim to an invention for a computer program, so can a claim to steps leading to an answer be a claim to an invention for a performing a mental act.[53]

He then applied his principle to the *Fujitsu* case and concluded that a claim to a computer operating in a particular way was no more patentable than a claim to a computer program, and, therefore, a claim to a method of carrying out a calculation (a method of performing a mental act) was no more patentable when claimed as being done by a computer than when done on a piece of paper.[54]

From case law above, it can be concluded that computer programs that fall within one of the following categories are not patentable.

(1) Computer programs as such;
(2) Computer programs which produce a technical effect but such effect resides in computer programs;
(3) Computer programs which produce a technical effect but such effect is equivalent to a method of doing business;
(4) Computer programs which produce a technical effect but such effect is equivalent to a method of performing mental acts.

---

[51] *Fujitsu*, supra note 28.
[52] *Wang Laboratories Inc.'s Application* [1991] R.P.C. 463.
[53] Ibid., at 472.

Therefore, the only channel left for a patent protection for computer programs is that a claim has to be directed to a technical effect which is produced by computer programs and that the effect must exist outside computer programs themselves. It follows that the impact of the proposed framework on UK patent law has to be assessed on this basis. The current situation is that only a handful of computer programs are patentable (most computer programs in the market are left to be protected under copyright law). Thus, it is to be seen in the analysis in the next section whether the proposed framework is likely to bring any change to UK software patent law.

4. IMPACT ON THE CURRENT POSITION OF UK PATENT LAW

As discussed in Chapter four, once the proposed framework is established, the software company, which holds copyright in computer programs, will not be able to use legal tactics to prohibit access to ideas and principles underlying its computer programs. A copyright infringement would solely be determined from substantial similarity between two end products. The effect of the proposed framework is that competition in the software market will be stimulated. More small and medium-sized enterprises (SMEs) will be able to enter the market without fear of a legal suit from leading software companies in that market.

Clearly, such leading software companies will foresee that if there is more efficient competition in the market, they may lose some part of their share in that market. Quite naturally, they will seek some other forms of legal protection in order to protect the ideas and principles underlying the programs. One form of legal protection is, of course, patent law. The main theoretical reason why software companies are attracted by patent protection is that patent gives a monopoly right to the inventor in the sense that a defence of originality cannot be used to avoid patent infringement. It is this sort of power which the holder of patents can use to sue the infringer and which is not given by copyright. In practice, software companies seek to obtain a patent because they can use it for cross-licensing. For example, Oracle has been forced to protect itself by selectively applying for patents which will present the best opportunities for

---

[54] *Fujitsu*, supra note 28, at 621.

cross-licensing between Oracle and other companies who may allege patent infringement.[55] For these reasons, leading software companies have been attempting to file applications for patents for their computer programs in the form of SRI. For instance, in the US Microsoft had only 14 patents on SRI in 1992[56] but it now has more than 500 SRI patents.[57]

As seen from the above discussion on the current UK situation, patents for SRI are available only to a limited extent. Therefore, it is expected that there will be a pressure from the leading software companies for a move towards broader software patentability. They may even argue further for the abolition of computer programs from the list of non-patentable matters. Therefore, it can be said that an impact on patent law will be a move towards broader software patentability than is currently the position. This may result in pressure on Parliament to amend the Patents Act 1977 in favour of a broader and clearer provision of section 1. This is to create legal certainty and to cope with the demand for substitute protection.

Such an attempt may firstly be seen in the form of an argument against the authorities established by the Patents Court and the Court of Appeal. For example, the judgment of Falconer J. in *Merrill Lynch's Application* will be discounted because his principle (if the inventive step resides in the excluded matter, the invention is not patentable), leads to a misleading interpretation of the Patents Act 1977. It also distorts the intention of the Act which does not allow a broad interpretation of the excluded-matter. The judgment of Purchas L.J. in *Genentech* will lend support to such an argument. He stated:

> any of the matters listed in sub-paragraphs (a) to (b) shall not be an invention for the purposes of the Act ... and shall only to the extent that the application or patent relates to that step as such be disqualified.[58]

---

[55] Patent policy of Oracle Corporation, <http://lpf.ai.mit.edu/Patents/testimony/statements/oracle.statement.html> accessed on 3/12/00

[56] Search at <http://www.uspto.gov/patft/index.html> and <http://128.109.179.23/access/search-bool.html> by using keywords "Microsoft" and "program", accessed on 6/12/00. This database is to be discontinued on 31 December 2000.

[57] <http://www.ece.utexas.edu/courses/fall_99/ee302/homework/solutions/hw2_soln.html> accessed on 6/12/00.

Moreover, the judgment of Fox L.J. in the Court of Appeal in *Merrill Lynch* will also be partly criticised in that his Lordship did not explain clearly why the reasoning of Falconer J. was wrong. Instead, Fox L.J.'s dismissal of the appeal was founded on a different basis, namely, the invention was merely claimed to be a method of doing business. Thus, the absence of the Court of Appeal's clear disapproval of the judgment of Falconer J. could be deemed that the Court of Appeal agreed with Falconer J.. Such an assumption would be heavily criticised by the leading software companies.

Indeed, the judgment of Falconer J. was severely criticised by the Court of Appeal in *Genentech Inc.'s Patent*. Therefore, it is predictable that in the near future the judgment of the Court of Appeal in the *Genentech* case and certain parts of the Court of Appeal's judgments in the *Fujitsu* case and the *Merrill Lynch* case would be used by the leading software companies as a foundation for arguing for broader software patentability

Although the major part of judgment of the Court of Appeal in *Merrill Lynch* was criticised, some parts of its judgment can still be used by the leading software companies for the extension of software patentability.

> As I have already indicated, Article 53 of the Convention, like section 1(2), excludes from patentability mathematical methods as such. The Board held (paragraph 6) [in the *Vicom* case] that even if the idea underlying an invention is considered to reside in a mathematical method, a claim directed to a technical process in which the method is used does not seek protection for the mathematical method as such.[59]

Similarly, certain parts of the judgment of Aldous L.J. in the *Fujitsu* case will also play an important role in the argument for broader software patentability, i.e. the substance of the claimed invention is more important than the form.

> Fox L.J. was making it clear that it was not sufficient to look at the words of the claimed monopoly. The decision as to what was patentable depended

---

[58] *Genentech*, supra note 35, at 207.

upon substance not form. He also went on to point out the importance of considering whether the invention made a technical contribution, despite the fact that neither the statute nor Article 52 of the Convention lays down that the matter, which would result in the invention not relating to the thing as such, must provide a technical contribution.[60]

Therefore, it is predictable that the impact of the proposed framework on patent law would be a move towards the broadening of the current interpretation of UK patent law. This may develop further to the point that the abolition of computer programs from the list of non-patentable matters is required. Along the line of such development, there will be an attempt to argue against some authorities established by the Patents Court and the Court of Appeal which have a policy suggesting a narrow approach of software patentability.

On the other hand, companies, which are fearful of an absolute monopoly provided by patent law, may argue for the creation of a new kind of protection for computer programs, e.g. the utility model,[61] as is used by some countries in Europe, such as Germany and France. The reasoning of these companies may be that patent law provides protection for inventions up to 20 years. During that period these companies may not make use of the patented ideas and by the time the patent expires, the knowledge that becomes freely available will be of little value. By protecting computer programs under the utility model, the length of protection would be reduced to around 6-10 years only. This would give these companies a chance to make use of the computer programs after the protection has expired.

Indeed, companies, which produce patentable computer programs, may also seek an alternative protection from the utility model protection. This is because these companies may consider that the procedure of applying for a patent is too complicated and takes too long. The traditional period of obtaining a patent is between 2-3 years.[62] Although the Patent Office offers a speedier service, by combining the search and examination stages, which would take as little as one year, that service may not

---

[59] *Merrill Lynch* (CA), supra note 19, at 568.
[60] *Fujitsu*, supra note 28, at 614.
[61] The utility model protection is being proposed by the Commission, see discussion in section 5 of this chapter.

be appropriate for examining a large and complex computer program. As computer programs normally have a short commercial life, these companies will seek to obtain a swifter protection from the utility model. Therefore, it can be seen that another impact of the proposed framework on patent law would be an argument for the introduction of the utility model.

To answer whether or not the impact discussed above is justified, it must consider the fact that such an impact is or is not consistent with the development of UK patent law. The development of UK patent law in turn can be indicated from the signal given by the EPO, e.g. the decision of the technical Board of Appeal. Thus, in order to answer that question correctly, the position of software patentability under the EPC needs to be scrutinised and compared with the position of UK patent law. This is the subject of the next section.

5. CURRENT POSITION OF SOFTWARE PATENTABILITY UNDER EPC AND COMPARISON TO UK POSITION

The law of patent in the European Union is governed by the European Patent Convention (EPC).[63] As one of the signatories to the EPC, the United Kingdom has to provide basic requirements for patentability in accordance with Article 52 of the EPC. In this section, the position of software patentability under the EPC will be examined and compared to that under the UK Patents Act 1977. As will be seen from the discussion below, there are some differences in the approach between the Board of Appeal of the EPO and the UK court in determining software patentability, even

---

[62] Bainbridge, supra note 29, at 333.

[63] Signed in Munich in 1973, the EPC is the outcome of the European countries' collective political determination to establish a uniform patent system in Europe. The European Patent Organisation was established by the EPC. It comprises the legislative body, the Administrative Council, and the executive body, the European Patent Office (EPO). The EPO is not an EU institution but it is an international patent-granting authority. It is completely self-financing and has a large degree of administrative autonomy. The European Patent Organisation, for which the EPO acts as executive arm, currently has 19 member states: all the EU countries plus Cyprus, Liechtenstein, Monaco and Switzerland. About the EPO, <http://www.european-patent-office.org/epo_general.htm#office> accessed on 20/10/00. Detailed information about the EPO, <http://www.european-patent-office.org/epo_detailed.htm> accessed on 20/10/00.

though the wording of s. 1 of the Patents Act 1977 can be said to reflect the provisions of Article 52.[64]

Like the wording of s. 1(2) of the Patents Act 1977, Article 52 of the EPC does not grant a patent for a computer program as such. Software patentability, therefore, is dependent upon how the Board of Appeal interprets such provisions. The most prominent decision of the Board of Appeal, which sets the position of software patentability under the EPC apart from that under the Patents Act 1977, is the decision in *Vicom Systems Inc.'s Patent Application.*[65]

In this case, the Board of Appeal had to consider whether the methods for digitally processing images were merely mathematical methods or technical processes carried out in accordance with a mathematical algorithm. The Board of Appeal differentiated between a mathematical method and a technical process as follows:

> A basic difference between a mathematical method and a technical process can be seen, however, in the fact that a mathematical method or a

---

[64] As one of the signatories to the EPC, the United Kingdom has to provide basic requirements for patentability in accordance with Article 52 of the EPC.

> Article 52: Patentable inventions ...
> (2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1:
> (a) discoveries, scientific theories and mathematical methods;
> (b) aesthetic creations;
> (c) schemes, rules and methods for performing mental acts, playing games or doing business, and **programs for computers**;
> (d) presentations of information.
> (3) The provisions of paragraph 2 shall exclude patentability of the subject-matter or activities referred to in that provision only to the extent to which a European patent application or European patent relates to such subject-matter or activities **as such**. ...
> **(emphasis added)**.

Like the wording of s. 1(2) of the Patents Act 1977, Article 52 of the EPC does not grant a patent for a computer program as such. Software patentability, therefore, is dependent upon how the Board of Appeal interprets such provision.

[65] VICOM/Computer-related invention, Case number T208/84 (OJ 1987, 14), <http://www.law.soton.ac.uk/internal/llb2000-01/year3/infotechlw319/VICOMfulltext.htm> accessed on 17/10/00. The claimed invention in *Vicom* was a method of digitally processing images and an apparatus for carrying out the method. This involved scanning the elements of a data array to produce a convolved array and then repeating the scanning so that the last convolved data array generated had the enhanced qualities required. The application was rejected by the Examining Division for two reasons. Firstly, the method of digitally filtering a two-dimensional data array (representing a stored image) was a mathematical method. This was because the characterising part of the claim would only add a different mathematical concept and would not define new technical subject-matter in terms of technical features. Secondly, the implementation of the claimed methods for image processing by a

mathematical algorithm is carried out on numbers (whatever these numbers may represent) and provides a result also in numerical form, the mathematical method or algorithm being only an abstract concept prescribing how to operate on the numbers. No direct technical result is produced by the method as such. In contrast thereto, if a mathematical method is used in a technical process, that process is carried out on a physical entity (which may be a material object but equally an image stored as an electric signal) by some technical means implementing the method and provides as its result a certain change in that entity. The technical means might include a computer comprising suitable hardware or an appropriately programmed general purpose computer.[66]

The Board of Appeal went on:

> ... even if the idea underlying an invention may be considered to **reside in** a mathematical method a claim directed to a technical process in which the method is used does not seek protection for the mathematical method as such.[67] **(emphasis added)**

As the application for a patent was directed to the technical features of the invention, ' the Board of Appeal held that the claim was not barred from protection by Article 52(2)(a) and (3) EPC.[68] It is clear from the reasoning of the Board of Appeal that a patentable invention can reside in a mathematical method, provided that a claim is drafted in terms of the technical process of the invention.

By the same token, the Board of Appeal allowed a claim which was directed to a technical process or effect that resided in a computer program. It was held that:

> Generally, claims which can be considered as being directed to a computer set up to operate in accordance with a specified program (whether by means of hardware or software) for controlling or carrying out a technical process

---

program run on a computer was a claim directed to a computer program as such. Therefore, it could not be regarded as an invention under Article 52(2)(c) and (3) EPC (at para. 2 and 10).

[66] Ibid., at para. 5.

[67] Ibid., at para. 6.

[68] Ibid., at para. 9.

> cannot be regarded as relating to a computer program as such and thus are not objectionable under Article 52(2)(c) and (3) EPC.[69]

According to the approach of the Board of Appeal above, it can be seen that a patent can be granted for a technical effect produced by a computer program, regardless of whether or not the inventive step resides in the computer program itself. Nonetheless, it is necessary that the two following conditions are met. First, a patent application is not directly for the computer program itself. Second, other requirements for the technical effect to be patented are present, namely novelty, inventive step and industrial application.

The Board of Appeal also noted that 'it would seem illogical to grant protection for a technical process controlled by a suitably programmed computer but not the computer itself when set up to execute the control'.[70] This logical reasoning of the Board of Appeal establishes a principle which the Board of Appeal in subsequent cases uses as a basis for extending patent protection to computer programs.

In *Re IBM's Application*,[71] the Board of Appeal developed the principle established in *Vicom* to cover computer programs. It said:

> By analogy, the present Board finds it illogical to grant a patent for both a method and the apparatus adapted for carrying out the same method, but not for the computer program product, which comprises all the features enabling the implementation of the method and which, when loaded in a computer, is indeed able to carry out that method.[72]

Therefore, such an extension of the scope of software patentability will increase the number of patentable SRI. In comparison with the UK approach under the Patents Act 1977, it appears that the Board of Appeal tends to construe the excluded matters more narrowly than the way the UK court does. This will result in a broader scope of

---

[69] Ibid., at para. 15.
[70] Ibid., at para. 16.
[71] *Re IBM's Application*, decision of February 1999, Case number: T 0935/97 - 3.5.1; <http://www.european-patent-office.org/dg3/biblio/t970935eu1.htm> accessed on 1/12/99 and the document can be downloaded from <http://www.european-patent-office.org/dg3/pdf/t970935eu1.pdf>.
[72] Ibid., at para. 9.8.

software patentability under the EPC, in spite of the similarity in the wording of s. 1 of the Patents Act 1977 and of Article 52 of the EPC. It is submitted that such inconsistency is not desirable in the light of the current trend of legal harmonisation in the European Union.

It may be noted that in essence there is nothing in either the 1977 Act or the EPC which expressly allows technical effect to be patentable. The patentability of technical effect is derived mainly from the articulation of the provisions by case law. Therefore, the difference in those approaches originates from the methods employed by the UK courts and the Board of Appeal of the EPO in determining the scope of patentable technical effect.

In taking the narrow approach, the UK courts consider the question of the patentability of technical effect without the contribution of a computer program. This can be seen as a "segregative" method. The UK courts tend to construe the exclusion in s. 1(2)(c) broadly without taking into account the fact that some methods of performing a mental act cannot be carried out in reality.

In a relatively more generous approach, the Boards of Appeal of the EPO suggest that computer programs can be considered patentable if the application is directed to the process for which a patent is sought. Under the approach of the Boards of Appeal, a computer program in question does not have to be excluded from the invention. Thus, an invention can be patented even though the basic idea underlying the invention resides in the computer program itself.[73] This can be called an "integrative" method. The result of employing this method is that a patent may be granted where software manages, by means of a computer, an industrial process or the working of a piece of machinery. Moreover, where a program for a computer is the only means of obtaining a technical effect (such as one which is achieved by the internal functioning of a computer itself under the influence of a particular program), it can be patented.

This unique method of considering software patentability has been developed further to give effect to the patentability of a method of solving a technical problem. In *Re*

---

[73] *Re IBM's Application*, supra note 71, at para 7.4.

IBM's Application,[74] the Board of Appeal accepted that if the claimed invention could bring about the effect that had a technical character or caused software to solve a technical problem, that claimed invention would be considered as the subject-matter of a patent.[75] Thus, it can be seen that the scope of software patentability under the EPC has not only been extended to cover SRI whose technical effects reside in the computer program, but also to SRI which introduces a novel method of solving a technical problem by way of using computer programs.

To sum up, the approach of the Board of Appeal has resulted in a broader scope for patent protection for computer programs, compared with that of the UK. It can be said that computer programs are protectable under the EPC if they can produce a technical effect or provide a solution to a technical problem. It is immaterial whether or not the claimed innovative technical effect or solution is in essence residing in computer programs. Moreover, the patentable invention may involve a method of performing a mental or a mathematical method, so long as a claim is directed to the technical process of the invention. The approach of the Board of Appeal is, in effect, leading to the opposite result of the approach taken by the UK court. It is submitted that the inconsistency in the approach between the UK court and the Board of Appeal will lead to a change in the interpretation of the Patents Act 1977. This will be discussed below in section 6. It will also be answered in that section whether the impact of the proposed framework will obstruct or support the development of UK patent law.

6. ANALYSIS OF THE POTENTIAL DEVELOPMENT OF UK PATENT LAW

As can be seen from the discussion above, the current approach taken by the UK court is not consistent with that taken by the Board of Appeal of the EPO. The major difference between the UK court's approach and the Board of Appeal's approach is the way they consider the technical effect aspect of the claimed invention. If the technical effect resides in the computer program, the claim will be rejected under the UK court's approach whereas it will be allowed under the Board of Appeal approach.

---

[74] *Re IBM's Application*, supra note 71.

According to s. 130(7) of the Patents Act 1977, it is clear that the Patents Act 1977 has a harmonising objective. It is stated in this section that:

> ... by a resolution made on the signature of the Community Patent Convention the governments of the member states of the European Economic Community resolved to adjust their laws relating to patents so as to bring those laws into conformity with the corresponding provisions of the European Patent Convention ... it is hereby declared that the following provisions of this Act, this is to say, sections 1(1) to (4) ... are so framed as to have, as nearly as practicable, the same effects in the United Kingdom as the corresponding provisions of the European Patent Convention ...

Therefore, when construing and applying section 1(1) and (2) of the Patents Act 1977, the UK court must have regard to the legislative intention with which those subsections were framed.[76] For this reason, 'it is of the utmost importance that the interpretation given to section 1 of the Act by the courts in the United Kingdom, and the interpretation given to Article 52 of the European Patent Convention by the European Patent Office, should be the same.'[77] The provision of this section provides support for an argument that the UK court will in the near future have to change its approach to follow the lead taken by the Board of Appeals.

Although the recent judgments of the Court of Appeal do not show such conformity, it is likely that such desired harmonisation of the approach will be adopted in the UK soon. This can be seen from the *Genentech* case where the Court of Appeal rejected Falconer J.'s approach in *Merrill Lynch*, although the Court of Appeal in the *Merrill Lynch* itself was silent on Falconer J.'s principle. It can also be seen that attempts have been made by the Court of Appeal in several cases to apply the principle established by the Board of Appeal of the EPO.[78] However, it seems that the Court of Appeal often found it unhelpful and came to a somewhat contradictory conclusion.

---

[75] Ibid., at para 6.4.
[76] *Gale's Application*, supra note 41, at 322.
[77] Ibid., per Nicholls L.J.
[78] See the *Merrill* Lynch case, the *Genentech* case and the *Fujitsu* case, supra note 47, 35 and 28 respectively.

Nevertheless, this at least gives a sign of attempting to harmonise the approach of the UK court and the Board of Appeal.

A firmer sign of changing the approach can be said to come from the UK Patent Office. Recently, the Patent Office published a notice relating to claims to programs for computers under the Patents Act 1977.[79] In the notice, it states clearly that it will change its practice according to the Board of Appeal's decision in the *IBM* case.[80] As discussed above, the Board of Appeal was of the opinion that claims to a computer system when programmed were accepted if the program had the potential to produce a patentable technical contribution. The Patent Office also accepts the consequence which may occur from this decision, i.e. it would make no difference whether a program is claimed by itself or as a record on a carrier.[81] Thus, it is clear that the Patent Office is now willing to allow claims for inventions whose technical effects reside in computer programs.

Another factor which indicates the extension of the scope of software patentability are the provisions of the Trade Related Aspects of Intellectual Property Rights Agreement (TRIPs). The TRIPs Agreement provides in Article 27(1) that Member countries have to make patents available for any invention in all fields of technology without discrimination. Although the TRIPs Agreement does not apply directly to the European Patent Organisation as the European Patent Organisation itself is neither a member of the World Trade Organisation (WTO) nor a signatory to the TRIPs Agreement, the TRIPs Agreement does apply to the UK via the ratification of the European Community. Indeed, the Board of Appeal in the *IBM* case adopted the principle set out in the TRIPs Agreement for the purpose of global harmonisation of patent law. The Board of Appeal was of the opinion that:

> although TRIPs may not be applied directly to the EPC, the Board thinks it appropriate to take it into consideration, since it is aimed at setting common standards and principles concerning the availability, scope and use of trade-

---

[79] The Patent Office – Claims to Programs for Computers, <http://www.patent.gov.uk/snews/notices/practice/programs.html> (last updated 10 January 2000) accessed on 13/10/00.
[80] *Re IBM's application*, supra note 71.
[81] The Patent Office, supra note 104, at para. 2-3.

> related intellectual property rights, and therefore of patent rights. Thus, TRIPs gives a clear indication of current trends [not to exclude computer programs from patentability]. ... In the Board's opinion they may contribute to the further highly desirable (world-wide) harmonisation of patent law.[82]

For this reason, it is submitted that the trend of UK patent law is to change its approach towards a broader scope of software patentability, i.e. to allow computer programs to be patented, at the very least, to the same extent as the current approach of the EPO (in the light of the Board of Appeal's decision in *Re IBM's application*). Therefore, it can be concluded that the impact of the proposed framework is consistent with the development of UK patent law. Thus, at this stage, it can be said that the proposition of this thesis is indeed justified and seems to support such a development of UK patent law.

However, the development of UK patent law is also subject to the forthcoming developments from the EU and the WIPO. It is necessary to observe and consider what these are, whether or not they provide a sound basis for changing the existing law and what is the implication of the proposed framework on such developments. The forthcoming developments from the EU and the WIPO are observed in the next section. Then, it will be followed, in section eight, by a comment on the appropriateness of such developments and how the proposed framework will have an implication on it.

7. FORTHCOMING DEVELOPMENT FROM EU AND WIPO

This section observes the current move at the European Union and the international levels which may have effect on UK patent law in the future.

---

[82] *Re IBM's Application*, supra note 71, at para 2.3 and 2.6. Effectively, the TRIPs agreement requires Member States to introduce intellectual property statutes as the price for benefiting from the free trade provisions of the General Agreement on Tariffs and Trade (GATT); see Ian J Lloyd, *Information Technology Law 3rd ed.* (Butterworths: London, 2000) p. 314.

7.1 Proposal for the revision of the EPC

On 24 February 2000, the Administrative Council of the European Patent Organisation advised that a conference of the contracting states should be convened to revise the EPC.[83] This conference was held in Munich between 20-29 November 2000. This potential change or modification of the EPC might be said to stem from a fierce competition in the software market between the United States and the European Union. While SRI can be protected, to a greater extent, under patent law in the United States, they are not protected to the same extent in Europe. The difference in the level of protection also affects the level of enforcement. In the United States, as it has become possible to lodge claims covering a program "as such", the holder of a patent covering a program may directly attack the distributor of counterfeit programs distributed via a medium (direct infringement). In contrast, in Europe, as the protection is limited to the technical invention which uses the program, the distributor of a diskette is only the accomplice, but not the author of the infringement (contributory infringement). The sole author of the infringement is the user who uses the program on the diskette and only he can be sued.[84]

Furthermore, because of the lack of harmonisation between the EPC and patent law in contracting states, most SMEs are not aware that patent protection can be obtained for SRI. As a result, around 75 percent of the patents for SRI in Europe are held by non-European companies.[85] Based on this statistic, the European Commission considered that the stimulation and advancement of software technology could be achieved by a wider and clearer scope of patent law. Therefore, it requested the EPO to take steps to modify Article 52.

In the Basic Proposal submitted on 13 October 2000, the Administrative Council proposed, *inter alia*, that "programs for computers" be deleted from Article 52(2)(c).[86] This is to bring the EPC into line with Article 27(1) of the TRIPs Agreement which

---

[83] Decision of the Administrative Council of 24 February 2000 convening a conference of the contracting states to revise the European Patent Convention, Article 1(1). <http://www.european-patent-office.org/epo/ca/e/24_02_00_e.htm> accessed on 18/10/00.
[84] Communication from the Commission, infra note 87, at 3.2.1.
[85] Ibid.
[86] Basic proposal for the revision of the European Patent Convention, done at Munich 13/10/2000, MR/2/00, p. 43 (explanatory remarks 3).

'requires Member countries to make patents available for any inventions, whether products or processes, in all fields of technology without discrimination, subject to the normal tests of novelty, inventiveness and industrial applicability. It is also required that patents be available and patent rights enjoyable without discrimination as to the place of invention and whether or not products are imported or locally produced'.[87]

Moreover, the Administrative Council considered that the EPO and the Board of Appeal had always interpreted and applied the EPC in such a way that it allowed a claimed invention whose subject-matter consisted of or included a computer program. Therefore, the proposed deletion of "programs for computers" would be consistent with the recent decision of the Board of Appeal in *Re IBM's application*.

The proposed revision of the EPC also took into account the criteria for patentability created by the Board of Appeal. These were the "technical effect" and/or "solution of a technical problem". Thus, it was clearly expressed in the proposed new wording of Article 52(1) that "European patents shall be granted for any inventions, **in all fields of technology, provided that they** are new, involve an inventive step and are susceptible of industrial application".

However, after the conference, it was concluded that the issue regarding computer programs had not been sufficiently thought through to be the right time to make a binding decision on them.[88] Further consultations with interested parties are needed. Therefore, the contracting states, in the mean time, agreed to maintain the current provisions of Article 52, save for the addition of the phrase "in all fields of technology, provided that they" to Article 52(1).[89]

By way of examining the development of patent law which may be affected by the proposed framework, this chapter will make some comments on this issue. This will be discussed in section 8.

---

[87] WTO: Overview: the TRIPs Agreement <http://www.wto.org/english/tratop_e/trips_e/intel2_e.htm#patents> accessed on 18/10/00.
[88] Conference Resolution, MR/22/00 e, <http://www.european-patent-office.org/epo/dipl_conf/pdf/em00022.pdf> accessed on 10/12/00.

7.2 The Community Patent

On 24 June 1997, the European Commission presented the Green Paper on the Community patent and the patent system in Europe.[90] The aim was to launch a broad discussion on the need to take new initiatives in relation to patents and to reflect on the nature and content of any such initiatives. It was unanimously agreed that the correct functioning of the internal market required twofold action: on the one hand, the introduction of a unitary system of patent protection and, on the other hand, various additional harmonisation measures to make the system more transparent and more effective.

On 19 November 1998, the European Parliament adopted its opinion, suggesting that consistent and effective Community legislation in the field of patents was a vital factor in promoting the competitiveness of enterprises in the EU. Therefore, it was not sufficient to harmonise the concrete provisions of national patent legislation; it was necessary to draw up a Community Regulation. According to the Parliament's suggestion, the Commission identified the most urgent issues for which it should submit proposals. Two of the priority issues were:[91]

(1) the Community patent
(2) the patentability of computer programs

The Commission proposed that the Community patent must be unitary throughout the Community, affordable and guarantee legal certainty. The Community patent would co-exist with the national patents and the European patent, at least for the transitional period. Inventors would remain free to choose the type of patent protection best suited to their need.[92] However, in principle, the provisions of the EPC, which apply to European patent applications, will be applicable to applications for the Community

---

[89] Act Revision the Convention on the Grant of European Patents (Munich, 29 November 2000), MR/3/00 Rev. 1 e, <http://www.european-patent-office.org/epo/dipl_conf/pdf/em00003a.pdf> accessed on 10/12/00.
[90] COM(97) 314 final, 24.6.97.
[91] Promoting innovation through patents – The follow-up to the Green Paper on the Community Patent and the Patent System in Europe, (COM(1999)42), para. 1.3, <http://www.europa.eu.int/comm/internal_market/en/intprop/indprop/8682en.pdf> accessed on 11/12/99.
[92] Proposal for a Council Regulation on the Community patent, COM(2000) 412 final, 1.8.2000.

patent. This is because the Commission proposed that the Community patent would have a symbiotic nature between the Regulation on the Community patent and that of the EPC and that the Community patent would have to accede to the EPC.[93]

With relevance to computer programs, the Green Paper revealed that the current legal environment covering SRI did not provide sufficient transparency and needed to be clarified. The Commission, therefore, proposed two actions.[94] Firstly, the Commission would present a draft Directive aiming at harmonising Member States' legislation on the patentability of computer programs. The Directive would ensure uniform application and interpretation of the new rules on the patentability of computer programs throughout the whole Community. Secondly, in parallel with this legal action, the contracting states to the EPC would need to take steps to modify Article 52(2)(c) EPC, in particular to abolish computer programs from the list of non-patentable inventions. This was to ensure harmony between the work carried out at Community level and that undertaken within the framework of the EPC.

As can be seen from the discussion above, the second action has been started by the Administrative Council of the European Patent Organisation while, until the present time, the first action has not been evident. However, the Commission proposed that the provisions of the EPC concerning such subjects as conditions of patentability would be applicable to the Community patent. This means that Community patents will be granted in accordance with the provisions of the EPC, both the conditions of patentability and the exceptions to patentability. Thus, any amendment to be made to the EPC will of course be applicable to the Community patent.

7.3 The protection of inventions by the utility model

In an attempt to harmonise intellectual property rights in the European Union, the European Commission proposed in its Green Paper of 19 July 1995 a Community

---

[93] Ibid., at 2.3.
[94] Communication from the Commission to the Council, the European Parliament and the Economic and Social Committee: The follow-up to the Green Paper on the Community Patent and the Patent System in Europe, <http://www.europa.eu.int/comm/internal_market/en/intprop/indprop/8682en.pdf> accessed on 5/10/00.

Directive[95]. This Directive will aim at harmonising the utility model which existed in some Member States[96] and introducing the utility model in Member States which did not have this system.[97] The utility model aims at providing legal protection for technical, small-scale improvements which give a product greater utility than it had before. It also aims to prevent distortion of competition, which at present causes difficulties to small and medium sized enterprises (SMEs) which try to innovate.[98]

Largely inspired by the provision of the EPC, the utility model resembles a patent in that the invention must be new. It must possess "novelty" and must display a measure of inventive achievement. It must also involve an "inventive step" even though the level of inventiveness required is not as great as that of patents.[99] This is the effect of the use of the word "very" in Article 6(1) which reads:

> For the purposes of this Directive, an invention shall be considered as involving an inventive step if it exhibits an advantage and, having regard to the state of the art, is not **very** obvious to a person skilled in the art. **(emphasis added)**

In addition, the most useful feature of the utility model is that it provides a quick and simple registration for qualified inventions. An applicant will have to wait only 6 months on average, compared with 2-4 years for a patent. This means that protection can be obtained more rapidly and cheaply, but that the protection conferred is less secure. The length of protection is limited to 6 years, with the extension for two further periods of 2 years.

In its initial proposal for a Directive, the Commission proposed that SRI be excluded from the utility model protection. This is because such inventions are currently protected either by patent (inventions relating to computer programs) or by copyright (computer programs as such).[100] However, the Committee on Legal Affairs and

---

[95] (COM(95) 370).
[96] For example, Germany, Spain, France.
[97] For example, United Kingdom, Sweden.
[98] Proposal for a European Parliament and Council Directive approximating the legal arrangements for the protection of inventions by utility model, (COM(97) 691) - part 1.
[99] Ibid., at part 4(a)(1).
[100] Ibid., at part 5 Article 4

Citizens' Rights suggested amendments to the initial proposal, one of which included the utility model protection for SRI.[101] Such a suggestion was approved by the European Parliament.

Having adopted the suggestion from the Committee and the Parliament, the Commission has submitted the amended proposal[102] which, *inter alia*, extends the scope of protection to SRI[103] and which specifies that protectable inventions can cover both products and processes.[104] Moreover, the amended proposal makes it possible for the Directive to cover games and toys.[105] However, it is awaiting the second reading of the Parliament.

7.4 The WIPO Patent Law Treaty

There has been a recent international move towards the harmonisation of the procedures for obtaining and maintaining a patent. On 1 June 2000, Member States of the WIPO adopted the Patent Law Treaty (PLT),[106] which opens for signature on 2 June 2000 and which will enter into force once it has been ratified by ten countries. The PLT seeks to reduce the cost of patent protection and to make the process of obtaining a patent more user friendly and widely accessible.[107] Under the PLT, the requirements and procedures for national and regional patent applications, and those for the Patent Co-operation Treaty (PCT) international application, will be harmonised.

---

[101] Report on the proposal for a European Parliament and Council Directive Approximating the legal arrangements for the protection of inventions by utility model (COM(97)0691-C4-0676/97-97/0356(COD)), Committee on Legal Affairs and Citizens' Rights, Rapporteur: Julio Anoveros Trias de Bes.

[102] Amended proposal for a European Parliament and Council Directive approximating the legal arrangements for the protection of inventions by utility model, (COM(1999) 309 final/2) 12.07.1999, (97/0356 (COD)).

[103] Ibid., Article 4 at p. 6 and Recital 13 at p. 12.

[104] Ibid., Article 6 at p. 15.

[105] Ibid.

[106] WIPO, Patent Law Treaty, PT/DC/47, <http://www.wipo.int/eng/iplex/wo_plt.htm> accessed on 06/10/00.

[107] WIPO, Press Release PR/2000/222: Patent Law Treaty is finalized <http://www.wipo.int/eng/pressrel/2000/p222.htm> accessed on 06/10/00.

However, the PLT does not, at this stage, seek to harmonise the substantive patent law as can be seen from the provision of Article 2(2) which reads as follows:

> Nothing in this Treaty or the Regulations in intended to be construed as prescribing anything that would limit the freedom of a Contracting Party to prescribe such requirements of the applicable substantive law relating to patents as it desires.

Therefore, the grant of a patent is still determined according to patent law of Contracting parties where a patent is designated.

8. SUGGESTED TREND OF PATENT LAW AND IMPLICATION OF THE PROPOSED FRAMEWORK TO THAT EFFECT

As can be seen from the observation in section 7, the move towards the broadest scope of software patentability signalled by the proposed deletion of computer programs from the list of non-patentable matters in the EPC, is the strongest impact that may have on the development of UK patent law. One of the reasons for such deletion is to have the scope of software patentability in the EU extended to that in the US. It is submitted that simply deleting the phrase "programs for computers" from Article 52(2)(c) and adding the phrase "in all fields of technology" to Article 52(1) may not yield the same effect of software patentability as that of the US. This is because the wordings of the US patent statute and the EPC are framed differently. Section 101 of US patent code (35 U.S.C. § 101) provides that:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or **any** new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title. (emphasis added)

From this provision, it can be seen that the scope of patentable inventions in the US is very broad. To be qualified for a patent, the claimed invention merely has to be new and useful. Moreover, the four categories within which the claimed invention has to fall have been stretched, by the repetitive use of the expansive term "any", to "include

anything under the sun that is made by man", according to the Committee Reports[108] which have subsequently been acknowledged by the Supreme Court.[109] In comparison, patentable inventions under the EPC and the Patents Act 1977 are more restrictive. They have to be new, involve an inventive step and susceptible of industrial application. Therefore, it has undoubtedly been much easier to interpret the US statute to cover computer programs because the inventor has to prove only that his computer program is new and useful.

Supporting the above argument is the fact that the Supreme Court has identified that only three categories are unpatentable, namely "laws of nature, natural phenomena and abstract ideas".[110] Although computer programs, which are considered as some form of mathematical algorithms, are unpatentable abstract ideas, the Supreme Court explains that they are not so considered if they are reduced to some type of practical application, i.e. a useful, concrete and tangible result.[111] Such a very broad interpretation of the exception leaves room for subsequent courts to allow most computer programs to be patented. The Court of Appeals in *re Alappat*,[112] for example, applied such an interpretation and held that data, which was transformed by a machine through a series of mathematical calculations to produce a smooth waveform display on a rasterizer monitor, constituted a practical application of a mathematical algorithm because it produced a useful, concrete and tangible result (i.e. the smooth waveform). Therefore, it can be seen that even though the exception of computer programs is deleted from Article 52 EPC, the scope of software patentability will not be as broad as that of the US. This is because, under US patent law, the construction of the requirements of patentable inventions and the interpretation of the "abstract ideas" exception do help the inventor to show only that his or her computer program is useful.

With regard to "business method", the EU and the UK will find it more difficult to extend patent law to cover "business method" as has been done in the US. This is

---

[108] The Reports accompanying this patent code spell out the Congress' intent, S. Rep. No. 82-1979 at 5 (1952); H.R. Rep. No. 82-1923 at 6 (1952).
[109] *Diamond v. Chakrabarty*, 447 U.S. 303, 309 (1980); *Diamond v. Diehr*, 450 U.S. 175, 182 (1981).
[110] *Diamond v. Diehr*, ibid., at 185.
[111] Ibid.
[112] *Re Alappat*, 33 F.3d 1526 (Fed. Cir. 1994).

mainly because the concept of the "business method" exception in the US has no statutory basis. It is a judicially-created exception which has now been considered as an ill-conceived exception by the Court of Appeals in *State Street Bank & Trust Co. v. Signature Financial Group Inc.*[113] After having reviewed the cases which discussed the "business method" exception,[114] the Court of Appeals concluded that in fact none of those cases had the claims been rejected solely upon the basis of being a business method. The Court has always applied other exceptions, which have a clearer concept,[115] in rejecting the claims. Therefore, the Court has put an end to the "business method" exception.[116] As "business method" in the EU and the UK has a statutory basis and it has never been considered as an inappropriate exception, it is unlikely that the Court would extend patent protection to cover business methods. Such an extension can only be done by deleting the phrase "methods for doing business" from Article 52(2)(c) EPC.

As for software patentability, the attempt to extend the scope of patent law to cover all kinds of computer programs that meet the patenting criteria[117] needs to be considered carefully. The issue is whether or not such an extension will undermine the philosophy of patent law or has adverse effects on economy, especially in the EU.

The philosophy of patent law has been variably explained.[118] For example, Reid said that 'the state (or its personification) grants the inventor an exclusive monopoly for a

---

[113] *State Street Bank & Trust Co. v. Signature Financial Group Inc.*, 149 F.3d 1368 (Fed. Cir. 1998).
[114] *Re Howard*, 394 F.2d 869 (Fed. Cir. 1968); *re Schrader*, 22 F.3d 290 (Fed. Cir. 1994); *re Alappat*, ibid.; *re Maucorps*, 609 F.2d 481 (Fed. Cir. 1979); *re Meyer*, 688 F.2d 789 (Fed. Cir. 1982); *Hotel Security Checking Co. v. Lorraine Co.*, 160 F.467 (2d Cir. 1908).
[115] Ibid., at 1374-75.
[116] The *State Street* decision has resulted in a draft bill of the Business Method Patent Improvement Act 2000 which aims to reduce the risk of the USPTO awarding any more patents on the patently obvious. This Act will not prohibit the grant of business method patents but ensure that these kinds of patents will be issued when they truly represent something new and innovative. See, The Bureau of National Affairs Inc., Washington D.C., 'Bill Would Tighten PTO Procedure For Issuing Business Method Patents' (2000) 1 Computer Technology Law Report 198 (Number 11, 6 October 2000) <http://subscript.bna.com/SAMPLES/ctl.nsf/a190caa77300097c8525648000515ca5/6e7cb99b8b9b2b228525696f007f7c79?OpenDocument> accessed on 6/12/00.
[117] According to section 1 of the Patents Act 1977, the invention must be new, involves an inventive step and capable of industrial application.
[118] Fritz Machlup, *The Political Economy of Monopoly: Business, Labor and Government Policies* (The Johns Hopkins Press: Baltimore, 1952) p. 280-282; Peter Meinhardt, *Invention Patents & Trade Marks* (Gower Press Ltd.: London, 1971) p. 19 and 25-26; F. Liebesny, *Mainly on Patents* (Butterworths: London, 1972) p. 1; W. R. Cornish, *Intellectual Property 3rd ed.* (Sweet & Maxwell: London, 1996) p. 108-116.; David Bainbridge, *Intellectual Property 4th ed.* (FT, Pitman Publishing: London, 1999) p. 321-326.

limited time in his new invention in return for his disclosure of the invention so that the public at large will be able to practise the invention once the patent expires'.[119] White was of the opinion that 'it is desirable in the public interest that industrial techniques should be improved. In order to encourage improvement, and to encourage also the disclosure of improvements in preference to their use in secret, any person devising an improvement in a manufactured article, or in machinery or methods for making it, may upon disclosure of his improvement at the Patent Office demand to be given a monopoly in the use of it for a period of sixteen years'.[120] In the United States, the philosophy of US patent law is rooted in Article I s. 8 of the US Constitution which states 'Congress shall have power to promote the progress of science and useful arts by securing for limited times to Authors and Inventors the exclusive right to their respective writings and discoveries'.

Having reviewed various schools of thought, it seems that the commonly accepted philosophy is that patent law serves as an instrument for promoting competition which will lead to advancement of technical progress and the growth of industry. To achieve this aim, the following theories on which patent law is based have to be accomplished. These theories are as follows.

1. ***Incentive to create theory***: the system of patent law should act as an incentive to create a new invention and to stimulate further exertion to achieve a greater invention.
2. ***Encourage to disclose theory***: patent law should induce an inventor to disclose his discoveries instead of keeping them a secret.
3. ***Reward theory***: the inventor should be rewarded in return for his disclosure of the invention and for the expense of developing the invention to the state at which it is commercially practical.
4. ***Temporary protection theory***: patent law should provide a temporary monopoly right to exploit the invention and to hold off competition.

---

[119] Brian C. Reid, *Intellectual Property Guides: A Practical Guide to Patent Law 3rd ed.* (Sweet & Maxwell: London, 1999) p. 3.
[120] T. A. Blanco White, *Patents for Inventions and the Protection of Industrial Designs 4th ed.* (Stevens & Sons: London, 1974) p. 1.

5. ***Benefit to public's practice theory***: patent law should make it possible for the public at large to be able to practice the invention once the patent expires.
6. ***Natural property right theory***: the grant of a patent is considered as a recognition by the State of the natural right of the inventor whose ideas should not be usurped or stolen by others.

In developing patent law, especially in extending the scope of patent law, the theories mentioned above have to be taken into account. As for computer programs which are the current controversial issue, the question is: if patent law is to extend to cover all kinds of computer programs, will the above theories be accomplished? If the answer is in the negative, it means that the extension of patent law will not actually promote technical progress and industrial growth. The extension of patent law, therefore, will not be desirable. However, if the analysis shows that all of the theories are followed, it will be appropriate to extend patent law to cover computer programs as such. As mentioned at the beginning of this chapter, computer programs have now been applied to assist business methods in electronic commerce over the Internet. A question also arises as to whether a monopoly given by patent law should be granted to such an e-commerce business method in the EU. It may be noted that the business method in this context is driven by computer programs, i.e. its innovative aspect is caused by the use of computer programs. Therefore, suffice it to say that if computer programs are not allowed to be patented, the business method will not be patentable.

According to the current patent system, the fourth and the sixth theories seem to have been achieved because current patent law gives a monopoly right for a period of the maximum of 20 years and this is considered as a kind of a property right which can be seen as the State's recognition of the natural property right. Thus, only four theories will be analysed.

1. Incentive to create theory

The system of patent law is such that an exclusive monopoly for a limited time is granted to the inventor in return for his disclosure of the invention. It is believed that a monopoly right granted to the inventor will act as an incentive to innovate. The relevant issue arising out of this system is whether or not a monopoly right will urge

other software companies to create new software inventions and will motivate the software company already holding a patent to develop a greater invention.

In the European software market, the majority of European software companies are classed as SMEs but US software companies are normally large firms. If a patent is to be granted for computer programs, it is likely that large firms, which are normally US companies, will be able to obtain patents on the majority of the state-of-the-art software technology. This is because small software companies do not usually have enough resources to devote to research and development (R&D) that large companies have.[121] It is also likely that large companies will be the first to acquire patents on some type of software which are regarded as the foundation of software technology. For example, Microsoft may be able to patent the entire Windows operating system which would make it extremely difficult for other software companies to develop a similar program. As the Windows operating system has already acquired a dominant sale on the personal computer, to develop a completely new operating system would need a large amount of investment in both R&D and marketing. This would not be feasible for SMEs to do so. Similarly, to allow methods for doing business in e-commerce to be patentable will prevent the development of software technology because the first company to secure a patent of a simple business method will be able to prevent others from developing further technology.[122] Therefore, extending the scope of patent law to computer programs and e-commerce business methods will in actual fact reduce incentive for SMEs to enter the market.

On the other hand, if SMEs are the first to enter the market, large companies may not be hindered by SMEs' patents. As it is possible to create a new invention around an existing patented invention, large companies would be able to use their stronger financial support to attract customers, e.g. via advertisement or other marketing methods, although they are the second or third comer in that market. An analogous

---

[121] Stanley Fischer and Rudiger Dornbusch, *Economics* (McGraw-Hill Inc.: New York, 1983) p. 223 quoting Joseph Schumpeter's argument.

[122] For example, Amazon was able to obtain a preliminary injunction from U.S. District Court Judge Marsha J. Pechman's, which barred New York-based Barnesandnoble.com from using its version of 1-Click technology, which allowed online shoppers to purchase goods with a single mouse click (*Amazon.com Inc. v. Barnesandnoble.com Inc.*, 73 F. Supp. 1228, 53 USPQ2d 1115 (W.D. Wash. 1999). News from CNET News.com on 2 December 1999 <http://news.cnet.com/news/0-1007-200-1476392.html?pt.salon> accessed on 6/12/00.

example can be seen from Microsoft's practice in the Internet browser market. Microsoft was using its strong financial support to get its Internet Explorer into the Internet browser program market even though it was the second comer after the dominance of Netscape Navigator.[123] Applying this example to a supposed situation, it could be that if Microsoft invented a "two click" business method of purchasing commodities over the Internet, it could have invested in advertising to persuade customers who wished to use this technology that it was better than Amazon's "one click" technology. Microsoft could then have become the leader in the market. Thus, Amazon's reasons for patenting the "one click" technology, namely that to protect the investment that has been put into developing this technology and to prevent attack by other companies who might one day be able to put it out of business, are not practically justified.[124]

Therefore, it can be concluded that with a much less strong financial support, SMEs would not have incentive to develop their software further. Eventually, SMEs would be left out from the market. This would lead to a situation whereby only large companies would have monopolies in the market. They would not see any need to innovate at a quick pace because there would be no threat of competition from SMEs. This accords with Fischer and Dornbusch's theory that granting a monopoly does not always increase the incentive to innovate.[125]

From the above scenario, it can be seen that in actual fact an incentive to innovate comes from two different directions as shown in Figure 1. Firstly, it comes from the system of patent law where companies see a monopoly provided by a patent as an opportunity to make profits out of their inventions. Secondly, it comes from the fact that companies feel a threat of competition so they need improve their product in

---

[123] Although the main legal protection for the Internet browser is copyright law, it is submitted that a similar situation would also occur under patent law because both of them are classified as industrial property rights.

[124] Jeff Bezos, CEO of Amazon, said the company spent thousands of hours and millions of dollars to develop the system. They subsequently patented the process because they did not want to see an innovation they spent time and money developing be adopted by their competitors. Amazon started applying for patents because they realised that they would be under attack by players who might well one day be able to put them out of business. "We don't want to be another Netscape," says Bezos, <http://cse.stanford.edu/classes/cs201/projects-99-00/software-patents/amazon.html> accessed on 6/12/00.

[125] Fischer and Dornbusch, supra note 121, at 223.

order to remain competitive. The incentive to innovate would create competition and competition would then lead to the creation of new innovations and the advancement of technology.

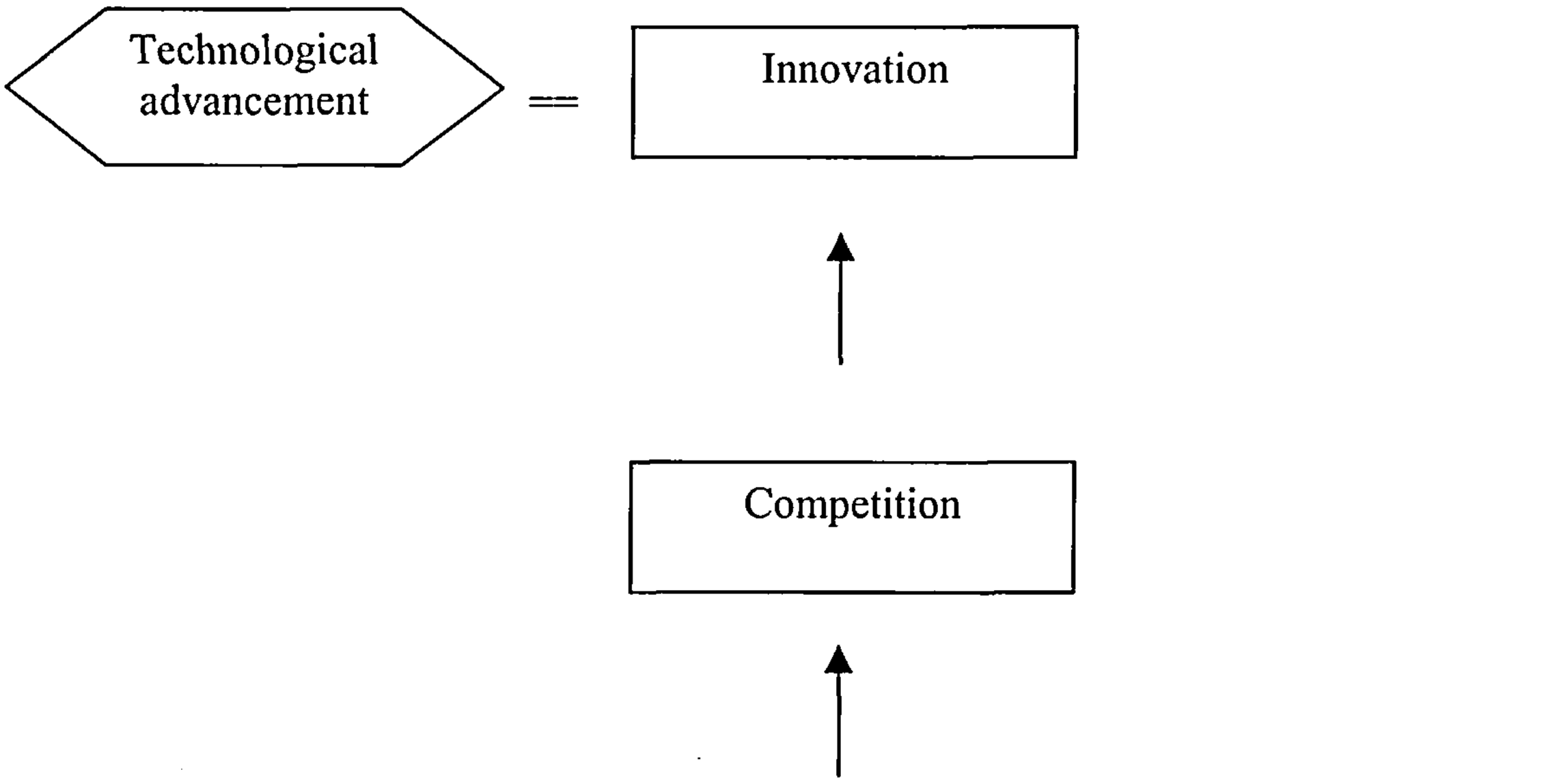


Figure 1

However, it is submitted that the benefit of receiving a monopoly in the software industry by the grant of patents does not create an incentive to innovate in the long run. This is because only large companies will be able to secure patents on the foundation of technology. Large companies can use their patents to prohibit other companies, especially SMEs from using the same technology. This will create a barrier to entry. It therefore means that the level of competition will decrease and large companies will not be pressured to improve their products quickly. Thus, when there is not enough competition in the market, the development of innovations will move at a slow pace. The public at large will not benefit from such a slow

development. On the other hand, if there remains a threat of competition, companies in the market, whether large or small, will feel the need to further develop their software. This will stimulate competition which would result in the development of new innovations and the growth of industry. Therefore, it is not the patent system that has to be promoted but the threat of competition that has to be preserved, as shown in Figure 2. Hence, the extension of patent law will not promote the incentive theory.

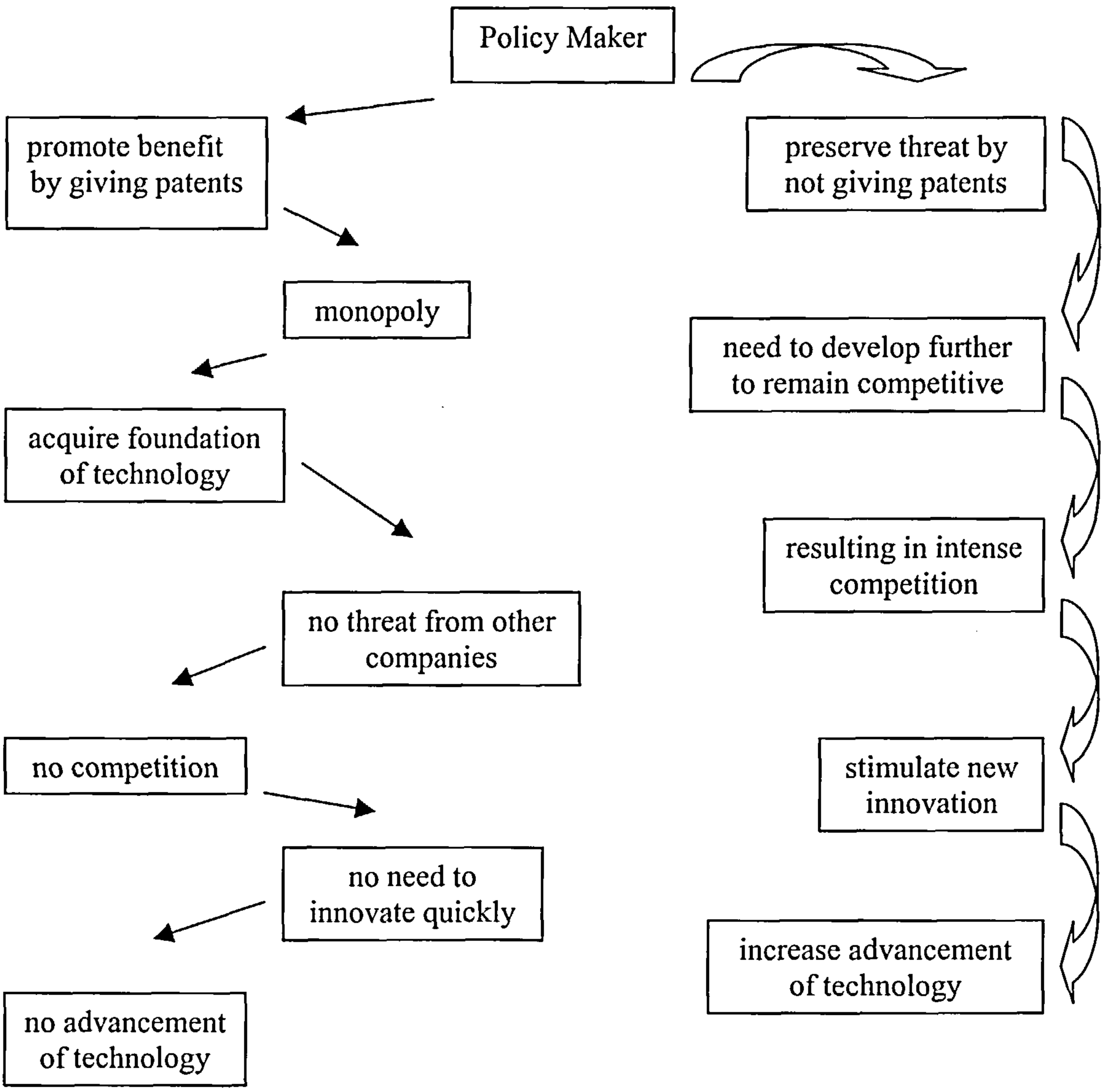


Figure 2

2. Encourage to disclose theory

One of the aims of patent law is to prevent the loss of knowledge after the death of the inventor. It has been reasoned that if inventors are not otherwise rewarded for their labours, they will try to get their reward by using their inventions in complete secrecy

and thus obtaining profits based on a monopoly in the knowledge of the secret. They may even die without having revealed their secret, so that it would be permanently lost to society.[126]

Based on this underlying assumption, it has to be assessed whether or not the patenting of software will prevent the loss of knowledge. To obtain a patent, the inventor is required to describe the claimed invention in accordance with section 14(5) of the Patents Act 1977.[127] This means that the specification must contain a description of the invention together with any drawings required to illustrate the invention.[128] As for computer programs in the form of SRI, the application would be framed in a flow diagram form describing the functions performed but it is not necessary to disclose the actual source code version of the programs.[129] Therefore, it may not be said that the patenting of computer programs will encourage a complete disclosure of computer programs to the public. As a result, the knowledge that is contained in the source code will actually be lost if the inventor chooses not to disclose it, even though it has been patented. Thus, it can be said that allowing computer programs to be patented does not fulfil the underlying assumption.

Moreover, the patenting of computer programs does not help achieve the global trend of encouraging compatibility among computer programs. Unless the disclosure of source code is to be made compulsory, companies wishing to create a program compatible with the patented program will find it difficult to create such a program even when the patent has expired. This is because, to be compatible, the new program has to have exactly the same interface code as the patented program. Although the advent of the Java programming technology which enables a program written in Java to be able to "run" on any platform, it does not fulfil the need for obtaining interface code in achieving compatibility between two or more application programs.[130]

---

[126] Machlup, supra note 118, at 281.
[127] Section 14(5): The claim or claims shall –
(a) define the matter for which the applicant seeks protection;
(b) be clear and concise
(c) be supported by the description; and
(d) relate to one invention or to a group of invention which are so linked as to form a single inventive concept.
[128] Bainbridge, supra note 29, at 328.
[129] Robert J. Hart kindly gave this explanation via e-mail contact on Friday 20 Oct 2000 11:33:26 EDT.
[130] See chapter 2 section 3.3.

Therefore, it can be said that allowing computer programs to be patented neither supports the software industry aim of compatibility nor fulfils the "encourage to disclose" theory that is one of the principles of patent law.

3. Reward theory

Under the patent system, a reward for the inventor is the right to use his invention in exclusion of others for a limited period. This right will prevent others from making, using, selling and importing the invention.[131] It is believed that the grant of a monopoly would enable the inventor to make a profit out of his invention by holding off competition.[132] However, the deliberate restraint of competition has a close link with the incentive theory in that the reward is used as an incentive to innovate. This will lead to the development of technology and the growth of industry which will improve the welfare of society. Thus, it can be said that the reward of granting a monopoly has the ultimate objective of serving the public interest.

In considering whether or not the extension of patent law to cover computer programs is appropriate, it has to be considered whether or not the reward will serve the public interest. It has been suggested that there are mainly three ways in which the reward, if given, would be at the detriment of the public interest.[133]

(a) when the control over the new technology is prolonged beyond the brief period contemplated by the law;
(b) when the patents through the way in which they are licensed to competitors are used to regulate competition among them and thereby to restrain competition far beyond the scope of the patent grant; and
(c) when individual companies are allowed to accumulate so many patents that they can control an important part of the technology of the industry..

As for point (a), it is possible in the field of software technology that the incremental improvement of patented computer programs (a mere extension of the first version)

---

[131] Section 60 of the Patents Act 1977.
[132] Meinhardt, supra note 118, at 19; Carl Chan, 'The Patentability of Software Data Structures After *Lowry* and *Warmerdam*' (1998) 32 New England Law Review 899.
[133] Machlup, supra note 118, at 282.

will enable the inventor to obtain a new patent on such an improved computer programs. As successful computer programs can create a "lock-in" effect, the continuation of securing patents for new versions of patented computer programs will ultimately create a permanent monopoly in the market. It is clear that this is not in the public interest. Furthermore, it may be considered that in fact the limited period of 20 years monopoly may not be appropriate for computer programs while it is justified for other kinds of inventions. It is very rare that a specific piece of software is used for 20 years before it is replaced by a newer version.[134] Most commercial software in the market nowadays is replaced with a newer version every 3-5 years.[135] For example, the Windows operating system has introduced a new version in approximately every three years (Windows 95, Windows 98 and Windows 2000); the Microsoft Word application program has also upgraded every 3 years (Word 6, Word 97 and Word 2000). Therefore, it can be inferred from the need of program upgrading that the commercial life of computer programs is much less than 20 years. Thus, it can be said that the 20-year monopoly is an over-reward for investment in the software industry. Also, the inventor of computer programs is usually able to recoup the investment and make profit from having the lead-time in the market. The inventor may not in fact need to rely on patent law for securing an attractive profit.

As for point (b), it is possible that software companies holding patents may use them in a prejudiced way to control competition in some sectors of the software market. However, in principle, such a malpractice is both controlled by the Patents Act 1977 itself[136] and competition law.[137] Thus, it is unlikely that the grant of patents would prejudice the public interest in this way.

As for point (c), however, it is likely that this may happen. This is because there is no limit on the number of patents allowed to be acquired for a single company. Therefore, it is possible that a large company would be able to obtain so many patents that it could control the main part of technology in a given software sector. This

---

[134] Nora M. Tocups and Robert J. O'Connell, 'Patent Protection for Computer Software' (1997) 14 The Computer Lawyer 18.
[135] According to accounting principles, software usually has the commercial life of three years. See, Michael Kavanagh, 'Accounting for dot.coms' (2000) Nov/Dec Accounting & Business 38.
[136] Section 46-54 (Licences of right and compulsory licences), section 70 (remedy for groundless threats of infringement proceedings) and section 71 (declaration or declarator as to non-infringement).

old English cases, is potent.[149] Therefore, the UK court may come to a different conclusion if it were to apply the fair dealing principle to the fact of this case.

5. CONCLUSION

From the discussion above, it can be seen that neither the current legal response in the United Kingdom nor that in the United States provides a proper solution to the commercial problems. It appears that the statutory responses, which can be found in the Software Directive and the CDPA 1988, focus mainly on two commercial problems, which are inability to develop interoperable products and inability to perform corrective maintenance, but ignore the remainders of the problems described at the beginning. Although these statutory responses attempt to solve only two commercial problems, the result is still unsatisfactory. This is not to mention the remainders of the problems which are arguably unsolved at all.

For the common law response which can be found in the United States, it cannot be said that this response provides a better solution to the commercial problems than the statutory response. This is because, until now, it has been tested against one commercial problem only, that is, inability to develop interoperable products. The result of the application of the common law response appears that there still be much uncertainty to apply the fair use doctrine to legalise reverse engineering. Therefore, its application to solve other commercial problems (such as inability to develop an idea-based product, inability to perform corrective, preventive and perfective maintenance) in the future is arguably unpredictable.

In the light of the current uncertainty and deficiency arising from the current legal response, this thesis suggests that a new approach to solve the commercial problems is required in order to provide a balance of interest between the copyright holder and the public. The proposed solution will be presented in the next chapter.

---

[148] Laddie, Prescott and Victoria, *The Modern Law of Copyright and Designs 2nd ed.* (Butterworths: London) Vol. 1 p. 91.
[149] Ibid.

patent. This is because the Commission proposed that the Community patent would have a symbiotic nature between the Regulation on the Community patent and that of the EPC and that the Community patent would have to accede to the EPC.[93]

With relevance to computer programs, the Green Paper revealed that the current legal environment covering SRI did not provide sufficient transparency and needed to be clarified. The Commission, therefore, proposed two actions.[94] Firstly, the Commission would present a draft Directive aiming at harmonising Member States' legislation on the patentability of computer programs. The Directive would ensure uniform application and interpretation of the new rules on the patentability of computer programs throughout the whole Community. Secondly, in parallel with this legal action, the contracting states to the EPC would need to take steps to modify Article 52(2)(c) EPC, in particular to abolish computer programs from the list of non-patentable inventions. This was to ensure harmony between the work carried out at Community level and that undertaken within the framework of the EPC.

As can be seen from the discussion above, the second action has been started by the Administrative Council of the European Patent Organisation while, until the present time, the first action has not been evident. However, the Commission proposed that the provisions of the EPC concerning such subjects as conditions of patentability would be applicable to the Community patent. This means that Community patents will be granted in accordance with the provisions of the EPC, both the conditions of patentability and the exceptions to patentability. Thus, any amendment to be made to the EPC will of course be applicable to the Community patent.

7.3 The protection of inventions by the utility model

In an attempt to harmonise intellectual property rights in the European Union, the European Commission proposed in its Green Paper of 19 July 1995 a Community

---

[93] Ibid., at 2.3.

[94] Communication from the Commission to the Council, the European Parliament and the Economic and Social Committee: The follow-up to the Green Paper on the Community Patent and the Patent System in Europe, <http://www.europa.eu.int/comm/internal_market/en/intprop/indprop/8682en.pdf> accessed on 5/10/00.

Directive[95]. This Directive will aim at harmonising the utility model which existed in some Member States[96] and introducing the utility model in Member States which did not have this system.[97] The utility model aims at providing legal protection for technical, small-scale improvements which give a product greater utility than it had before. It also aims to prevent distortion of competition, which at present causes difficulties to small and medium sized enterprises (SMEs) which try to innovate.[98]

Largely inspired by the provision of the EPC, the utility model resembles a patent in that the invention must be new. It must possess "novelty" and must display a measure of inventive achievement. It must also involve an "inventive step" even though the level of inventiveness required is not as great as that of patents.[99] This is the effect of the use of the word "very" in Article 6(1) which reads:

> For the purposes of this Directive, an invention shall be considered as involving an inventive step if it exhibits an advantage and, having regard to the state of the art, is not **very** obvious to a person skilled in the art. **(emphasis added)**

In addition, the most useful feature of the utility model is that it provides a quick and simple registration for qualified inventions. An applicant will have to wait only 6 months on average, compared with 2-4 years for a patent. This means that protection can be obtained more rapidly and cheaply, but that the protection conferred is less secure. The length of protection is limited to 6 years, with the extension for two further periods of 2 years.

In its initial proposal for a Directive, the Commission proposed that SRI be excluded from the utility model protection. This is because such inventions are currently protected either by patent (inventions relating to computer programs) or by copyright (computer programs as such).[100] However, the Committee on Legal Affairs and

---

[95] (COM(95) 370).
[96] For example, Germany, Spain, France.
[97] For example, United Kingdom, Sweden.
[98] Proposal for a European Parliament and Council Directive approximating the legal arrangements for the protection of inventions by utility model, (COM(97) 691) - part 1.
[99] Ibid., at part 4(a)(1).
[100] Ibid., at part 5 Article 4

Citizens' Rights suggested amendments to the initial proposal, one of which included the utility model protection for SRI.[101] Such a suggestion was approved by the European Parliament.

Having adopted the suggestion from the Committee and the Parliament, the Commission has submitted the amended proposal[102] which, *inter alia*, extends the scope of protection to SRI[103] and which specifies that protectable inventions can cover both products and processes.[104] Moreover, the amended proposal makes it possible for the Directive to cover games and toys.[105] However, it is awaiting the second reading of the Parliament.

7.4 The WIPO Patent Law Treaty

There has been a recent international move towards the harmonisation of the procedures for obtaining and maintaining a patent. On 1 June 2000, Member States of the WIPO adopted the Patent Law Treaty (PLT),[106] which opens for signature on 2 June 2000 and which will enter into force once it has been ratified by ten countries. The PLT seeks to reduce the cost of patent protection and to make the process of obtaining a patent more user friendly and widely accessible.[107] Under the PLT, the requirements and procedures for national and regional patent applications, and those for the Patent Co-operation Treaty (PCT) international application, will be harmonised.

---

[101] Report on the proposal for a European Parliament and Council Directive Approximating the legal arrangements for the protection of inventions by utility model (COM(97)0691-C4-0676/97-97/0356(COD)), Committee on Legal Affairs and Citizens' Rights, Rapporteur: Julio Anoveros Trias de Bes.

[102] Amended proposal for a European Parliament and Council Directive approximating the legal arrangements for the protection of inventions by utility model, (COM(1999) 309 final/2) 12.07.1999, (97/0356 (COD)).

[103] Ibid., Article 4 at p. 6 and Recital 13 at p. 12.

[104] Ibid., Article 6 at p. 15.

[105] Ibid.

[106] WIPO, Patent Law Treaty, PT/DC/47, <http://www.wipo.int/eng/iplex/wo_plt.htm> accessed on 06/10/00.

[107] WIPO, Press Release PR/2000/222: Patent Law Treaty is finalized <http://www.wipo.int/eng/pressrel/2000/p222.htm> accessed on 06/10/00.

However, the PLT does not, at this stage, seek to harmonise the substantive patent law as can be seen from the provision of Article 2(2) which reads as follows:

> Nothing in this Treaty or the Regulations in intended to be construed as prescribing anything that would limit the freedom of a Contracting Party to prescribe such requirements of the applicable substantive law relating to patents as it desires.

Therefore, the grant of a patent is still determined according to patent law of Contracting parties where a patent is designated.

8. SUGGESTED TREND OF PATENT LAW AND IMPLICATION OF THE PROPOSED FRAMEWORK TO THAT EFFECT

As can be seen from the observation in section 7, the move towards the broadest scope of software patentability signalled by the proposed deletion of computer programs from the list of non-patentable matters in the EPC, is the strongest impact that may have on the development of UK patent law. One of the reasons for such deletion is to have the scope of software patentability in the EU extended to that in the US. It is submitted that simply deleting the phrase "programs for computers" from Article 52(2)(c) and adding the phrase "in all fields of technology" to Article 52(1) may not yield the same effect of software patentability as that of the US. This is because the wordings of the US patent statute and the EPC are framed differently. Section 101 of US patent code (35 U.S.C. § 101) provides that:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or **any** new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title. (emphasis added)

From this provision, it can be seen that the scope of patentable inventions in the US is very broad. To be qualified for a patent, the claimed invention merely has to be new and useful. Moreover, the four categories within which the claimed invention has to fall have been stretched, by the repetitive use of the expansive term "any", to "include

anything under the sun that is made by man", according to the Committee Reports[108] which have subsequently been acknowledged by the Supreme Court.[109] In comparison, patentable inventions under the EPC and the Patents Act 1977 are more restrictive. They have to be new, involve an inventive step and susceptible of industrial application. Therefore, it has undoubtedly been much easier to interpret the US statute to cover computer programs because the inventor has to prove only that his computer program is new and useful.

Supporting the above argument is the fact that the Supreme Court has identified that only three categories are unpatentable, namely "laws of nature, natural phenomena and abstract ideas".[110] Although computer programs, which are considered as some form of mathematical algorithms, are unpatentable abstract ideas, the Supreme Court explains that they are not so considered if they are reduced to some type of practical application, i.e. a useful, concrete and tangible result.[111] Such a very broad interpretation of the exception leaves room for subsequent courts to allow most computer programs to be patented. The Court of Appeals in *re Alappat*,[112] for example, applied such an interpretation and held that data, which was transformed by a machine through a series of mathematical calculations to produce a smooth waveform display on a rasterizer monitor, constituted a practical application of a mathematical algorithm because it produced a useful, concrete and tangible result (i.e. the smooth waveform). Therefore, it can be seen that even though the exception of computer programs is deleted from Article 52 EPC, the scope of software patentability will not be as broad as that of the US. This is because, under US patent law, the construction of the requirements of patentable inventions and the interpretation of the "abstract ideas" exception do help the inventor to show only that his or her computer program is useful.

With regard to "business method", the EU and the UK will find it more difficult to extend patent law to cover "business method" as has been done in the US. This is

---

[108] The Reports accompanying this patent code spell out the Congress' intent, S. Rep. No. 82-1979 at 5 (1952); H.R. Rep. No. 82-1923 at 6 (1952).
[109] *Diamond v. Chakrabarty*, 447 U.S. 303, 309 (1980); *Diamond v. Diehr*, 450 U.S. 175, 182 (1981).
[110] *Diamond v. Diehr*, ibid., at 185.
[111] Ibid.
[112] *Re Alappat*, 33 F.3d 1526 (Fed. Cir. 1994).

mainly because the concept of the "business method" exception in the US has no statutory basis. It is a judicially-created exception which has now been considered as an ill-conceived exception by the Court of Appeals in *State Street Bank & Trust Co. v. Signature Financial Group Inc.*[113] After having reviewed the cases which discussed the "business method" exception,[114] the Court of Appeals concluded that in fact none of those cases had the claims been rejected solely upon the basis of being a business method. The Court has always applied other exceptions, which have a clearer concept,[115] in rejecting the claims. Therefore, the Court has put an end to the "business method" exception.[116] As "business method" in the EU and the UK has a statutory basis and it has never been considered as an inappropriate exception, it is unlikely that the Court would extend patent protection to cover business methods. Such an extension can only be done by deleting the phrase "methods for doing business" from Article 52(2)(c) EPC.

As for software patentability, the attempt to extend the scope of patent law to cover all kinds of computer programs that meet the patenting criteria[117] needs to be considered carefully. The issue is whether or not such an extension will undermine the philosophy of patent law or has adverse effects on economy, especially in the EU.

The philosophy of patent law has been variably explained.[118] For example, Reid said that 'the state (or its personification) grants the inventor an exclusive monopoly for a

---

[113] *State Street Bank & Trust Co. v. Signature Financial Group Inc.*, 149 F.3d 1368 (Fed. Cir. 1998).
[114] *Re Howard*, 394 F.2d 869 (Fed. Cir. 1968); *re Schrader*, 22 F.3d 290 (Fed. Cir. 1994); *re Alappat*, ibid.; *re Maucorps*, 609 F.2d 481 (Fed. Cir. 1979); *re Meyer*, 688 F.2d 789 (Fed. Cir. 1982); *Hotel Security Checking Co. v. Lorraine Co.*, 160 F.467 (2d Cir. 1908).
[115] Ibid., at 1374-75.
[116] The *State Street* decision has resulted in a draft bill of the Business Method Patent Improvement Act 2000 which aims to reduce the risk of the USPTO awarding any more patents on the patently obvious. This Act will not prohibit the grant of business method patents but ensure that these kinds of patents will be issued when they truly represent something new and innovative. See, The Bureau of National Affairs Inc., Washington D.C., 'Bill Would Tighten PTO Procedure For Issuing Business Method Patents' (2000) 1 Computer Technology Law Report 198 (Number 11, 6 October 2000) <http://subscript.bna.com/SAMPLES/ctl.nsf/a190caa77300097c8525648000515ca5/6e7cb99b8b9b2b228525696f007f7c79?OpenDocument> accessed on 6/12/00.
[117] According to section 1 of the Patents Act 1977, the invention must be new, involves an inventive step and capable of industrial application.
[118] Fritz Machlup, *The Political Economy of Monopoly: Business, Labor and Government Policies* (The Johns Hopkins Press: Baltimore, 1952) p. 280-282; Peter Meinhardt, *Invention Patents & Trade Marks* (Gower Press Ltd.: London, 1971) p. 19 and 25-26; F. Liebesny, *Mainly on Patents* (Butterworths: London, 1972) p. 1; W. R. Cornish, *Intellectual Property 3rd ed.* (Sweet & Maxwell: London, 1996) p. 108-116.; David Bainbridge, *Intellectual Property 4th ed.* (FT, Pitman Publishing: London, 1999) p. 321-326.

limited time in his new invention in return for his disclosure of the invention so that the public at large will be able to practise the invention once the patent expires'.[119] White was of the opinion that 'it is desirable in the public interest that industrial techniques should be improved. In order to encourage improvement, and to encourage also the disclosure of improvements in preference to their use in secret, any person devising an improvement in a manufactured article, or in machinery or methods for making it, may upon disclosure of his improvement at the Patent Office demand to be given a monopoly in the use of it for a period of sixteen years'.[120] In the United States, the philosophy of US patent law is rooted in Article I s. 8 of the US Constitution which states 'Congress shall have power to promote the progress of science and useful arts by securing for limited times to Authors and Inventors the exclusive right to their respective writings and discoveries'.

Having reviewed various schools of thought, it seems that the commonly accepted philosophy is that patent law serves as an instrument for promoting competition which will lead to advancement of technical progress and the growth of industry. To achieve this aim, the following theories on which patent law is based have to be accomplished. These theories are as follows.

1. ***Incentive to create theory***: the system of patent law should act as an incentive to create a new invention and to stimulate further exertion to achieve a greater invention.
2. ***Encourage to disclose theory***: patent law should induce an inventor to disclose his discoveries instead of keeping them a secret.
3. ***Reward theory***: the inventor should be rewarded in return for his disclosure of the invention and for the expense of developing the invention to the state at which it is commercially practical.
4. ***Temporary protection theory***: patent law should provide a temporary monopoly right to exploit the invention and to hold off competition.

---

[119] Brian C. Reid, *Intellectual Property Guides: A Practical Guide to Patent Law 3rd ed.* (Sweet & Maxwell: London, 1999) p. 3.
[120] T. A. Blanco White, *Patents for Inventions and the Protection of Industrial Designs 4th ed.* (Stevens & Sons: London, 1974) p. 1.

5. ***Benefit to public's practice theory***: patent law should make it possible for the public at large to be able to practice the invention once the patent expires.
6. ***Natural property right theory***: the grant of a patent is considered as a recognition by the State of the natural right of the inventor whose ideas should not be usurped or stolen by others.

In developing patent law, especially in extending the scope of patent law, the theories mentioned above have to be taken into account. As for computer programs which are the current controversial issue, the question is: if patent law is to extend to cover all kinds of computer programs, will the above theories be accomplished? If the answer is in the negative, it means that the extension of patent law will not actually promote technical progress and industrial growth. The extension of patent law, therefore, will not be desirable. However, if the analysis shows that all of the theories are followed, it will be appropriate to extend patent law to cover computer programs as such. As mentioned at the beginning of this chapter, computer programs have now been applied to assist business methods in electronic commerce over the Internet. A question also arises as to whether a monopoly given by patent law should be granted to such an e-commerce business method in the EU. It may be noted that the business method in this context is driven by computer programs, i.e. its innovative aspect is caused by the use of computer programs. Therefore, suffice it to say that if computer programs are not allowed to be patented, the business method will not be patentable.

According to the current patent system, the fourth and the sixth theories seem to have been achieved because current patent law gives a monopoly right for a period of the maximum of 20 years and this is considered as a kind of a property right which can be seen as the State's recognition of the natural property right. Thus, only four theories will be analysed.

1. Incentive to create theory

The system of patent law is such that an exclusive monopoly for a limited time is granted to the inventor in return for his disclosure of the invention. It is believed that a monopoly right granted to the inventor will act as an incentive to innovate. The relevant issue arising out of this system is whether or not a monopoly right will urge

other software companies to create new software inventions and will motivate the software company already holding a patent to develop a greater invention.

In the European software market, the majority of European software companies are classed as SMEs but US software companies are normally large firms. If a patent is to be granted for computer programs, it is likely that large firms, which are normally US companies, will be able to obtain patents on the majority of the state-of-the-art software technology. This is because small software companies do not usually have enough resources to devote to research and development (R&D) that large companies have.[121] It is also likely that large companies will be the first to acquire patents on some type of software which are regarded as the foundation of software technology. For example, Microsoft may be able to patent the entire Windows operating system which would make it extremely difficult for other software companies to develop a similar program. As the Windows operating system has already acquired a dominant sale on the personal computer, to develop a completely new operating system would need a large amount of investment in both R&D and marketing. This would not be feasible for SMEs to do so. Similarly, to allow methods for doing business in e-commerce to be patentable will prevent the development of software technology because the first company to secure a patent of a simple business method will be able to prevent others from developing further technology.[122] Therefore, extending the scope of patent law to computer programs and e-commerce business methods will in actual fact reduce incentive for SMEs to enter the market.

On the other hand, if SMEs are the first to enter the market, large companies may not be hindered by SMEs' patents. As it is possible to create a new invention around an existing patented invention, large companies would be able to use their stronger financial support to attract customers, e.g. via advertisement or other marketing methods, although they are the second or third comer in that market. An analogous

---

[121] Stanley Fischer and Rudiger Dornbusch, *Economics* (McGraw-Hill Inc.: New York, 1983) p. 223 quoting Joseph Schumpeter's argument.

[122] For example, Amazon was able to obtain a preliminary injunction from U.S. District Court Judge Marsha J. Pechman's, which barred New York-based Barnesandnoble.com from using its version of 1-Click technology, which allowed online shoppers to purchase goods with a single mouse click (*Amazon.com Inc. v. Barnesandnoble.com Inc.*, 73 F. Supp. 1228, 53 USPQ2d 1115 (W.D. Wash. 1999). News from CNET News.com on 2 December 1999 <http://news.cnet.com/news/0-1007-200-1476392.html?pt.salon> accessed on 6/12/00.

example can be seen from Microsoft's practice in the Internet browser market. Microsoft was using its strong financial support to get its Internet Explorer into the Internet browser program market even though it was the second comer after the dominance of Netscape Navigator.[123] Applying this example to a supposed situation, it could be that if Microsoft invented a "two click" business method of purchasing commodities over the Internet, it could have invested in advertising to persuade customers who wished to use this technology that it was better than Amazon's "one click" technology. Microsoft could then have become the leader in the market. Thus, Amazon's reasons for patenting the "one click" technology, namely that to protect the investment that has been put into developing this technology and to prevent attack by other companies who might one day be able to put it out of business, are not practically justified.[124]

Therefore, it can be concluded that with a much less strong financial support, SMEs would not have incentive to develop their software further. Eventually, SMEs would be left out from the market. This would lead to a situation whereby only large companies would have monopolies in the market. They would not see any need to innovate at a quick pace because there would be no threat of competition from SMEs. This accords with Fischer and Dornbusch's theory that granting a monopoly does not always increase the incentive to innovate.[125]

From the above scenario, it can be seen that in actual fact an incentive to innovate comes from two different directions as shown in Figure 1. Firstly, it comes from the system of patent law where companies see a monopoly provided by a patent as an opportunity to make profits out of their inventions. Secondly, it comes from the fact that companies feel a threat of competition so they need improve their product in

---

[123] Although the main legal protection for the Internet browser is copyright law, it is submitted that a similar situation would also occur under patent law because both of them are classified as industrial property rights.

[124] Jeff Bezos, CEO of Amazon, said the company spent thousands of hours and millions of dollars to develop the system. They subsequently patented the process because they did not want to see an innovation they spent time and money developing be adopted by their competitors. Amazon started applying for patents because they realised that they would be under attack by players who might well one day be able to put them out of business. "We don't want to be another Netscape," says Bezos, <http://cse.stanford.edu/classes/cs201/projects-99-00/software-patents/amazon.html> accessed on 6/12/00.

[125] Fischer and Dornbusch, supra note 121, at 223.

order to remain competitive. The incentive to innovate would create competition and competition would then lead to the creation of new innovations and the advancement of technology.

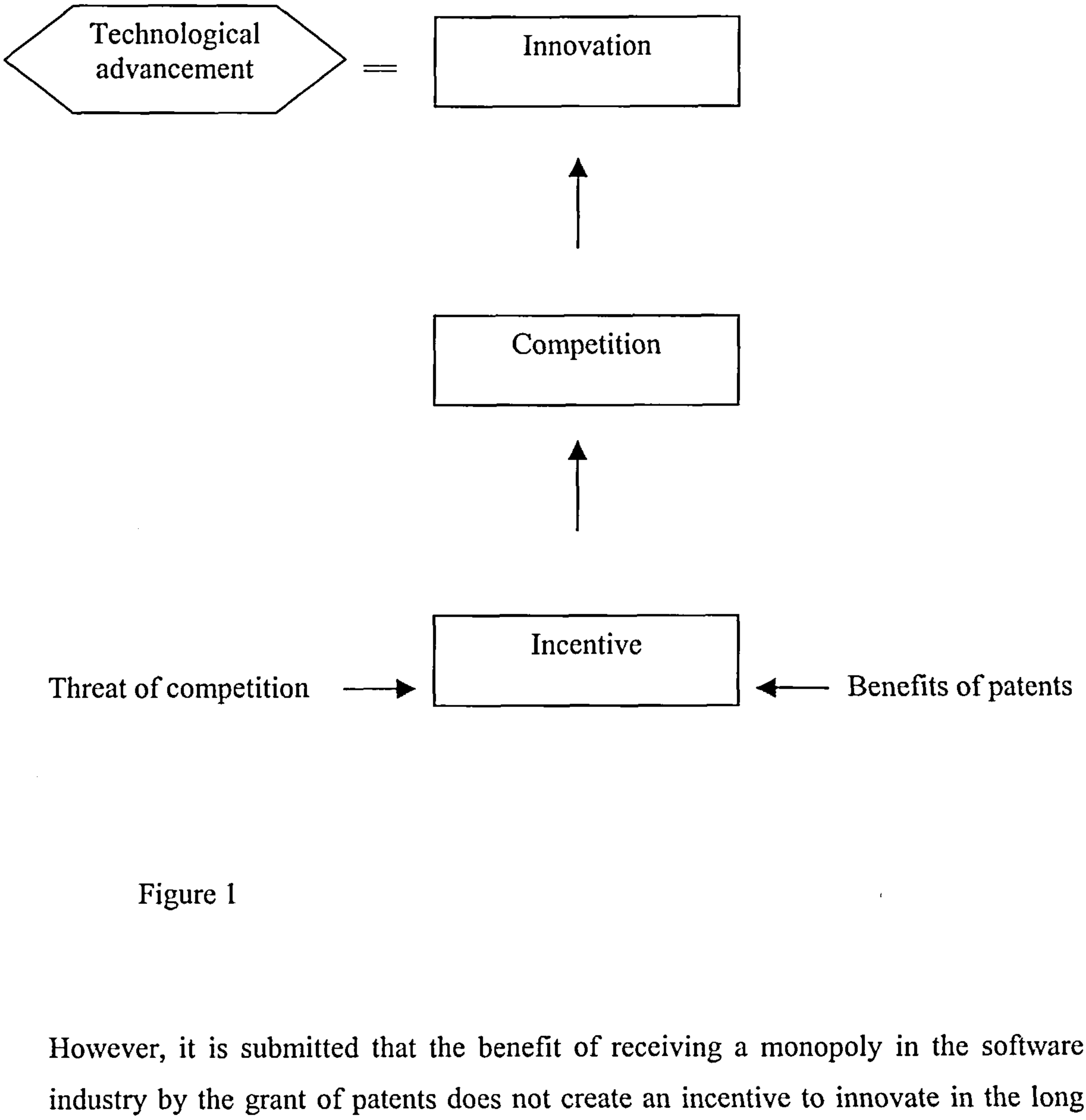


Figure 1

However, it is submitted that the benefit of receiving a monopoly in the software industry by the grant of patents does not create an incentive to innovate in the long run. This is because only large companies will be able to secure patents on the foundation of technology. Large companies can use their patents to prohibit other companies, especially SMEs from using the same technology. This will create a barrier to entry. It therefore means that the level of competition will decrease and large companies will not be pressured to improve their products quickly. Thus, when there is not enough competition in the market, the development of innovations will move at a slow pace. The public at large will not benefit from such a slow

development. On the other hand, if there remains a threat of competition, companies in the market, whether large or small, will feel the need to further develop their software. This will stimulate competition which would result in the development of new innovations and the growth of industry. Therefore, it is not the patent system that has to be promoted but the threat of competition that has to be preserved, as shown in Figure 2. Hence, the extension of patent law will not promote the incentive theory.

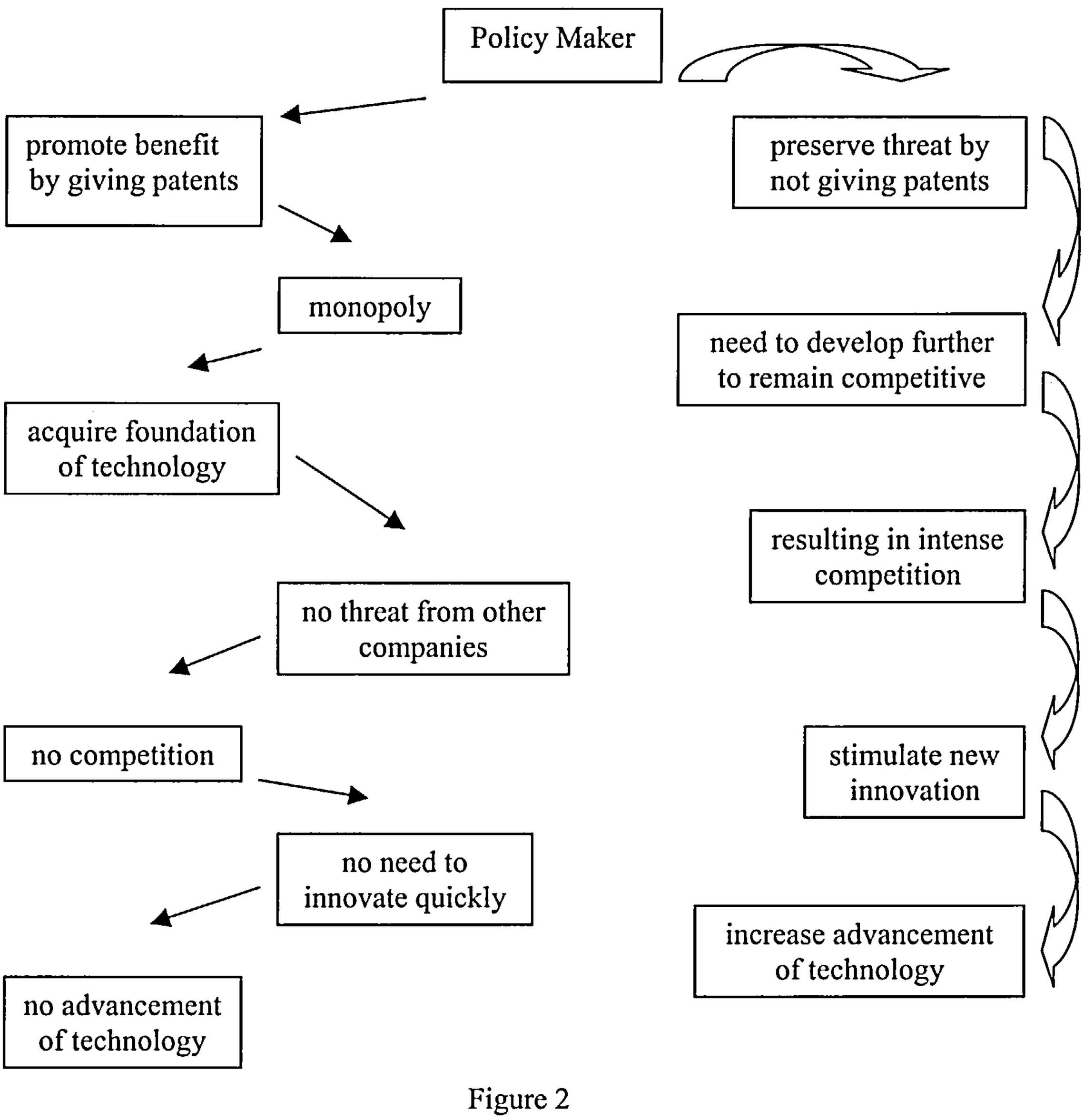


Figure 2

2. Encourage to disclose theory

One of the aims of patent law is to prevent the loss of knowledge after the death of the inventor. It has been reasoned that if inventors are not otherwise rewarded for their labours, they will try to get their reward by using their inventions in complete secrecy

and thus obtaining profits based on a monopoly in the knowledge of the secret. They may even die without having revealed their secret, so that it would be permanently lost to society.[126]

Based on this underlying assumption, it has to be assessed whether or not the patenting of software will prevent the loss of knowledge. To obtain a patent, the inventor is required to describe the claimed invention in accordance with section 14(5) of the Patents Act 1977.[127] This means that the specification must contain a description of the invention together with any drawings required to illustrate the invention.[128] As for computer programs in the form of SRI, the application would be framed in a flow diagram form describing the functions performed but it is not necessary to disclose the actual source code version of the programs.[129] Therefore, it may not be said that the patenting of computer programs will encourage a complete disclosure of computer programs to the public. As a result, the knowledge that is contained in the source code will actually be lost if the inventor chooses not to disclose it, even though it has been patented. Thus, it can be said that allowing computer programs to be patented does not fulfil the underlying assumption.

Moreover, the patenting of computer programs does not help achieve the global trend of encouraging compatibility among computer programs. Unless the disclosure of source code is to be made compulsory, companies wishing to create a program compatible with the patented program will find it difficult to create such a program even when the patent has expired. This is because, to be compatible, the new program has to have exactly the same interface code as the patented program. Although the advent of the Java programming technology which enables a program written in Java to be able to "run" on any platform, it does not fulfil the need for obtaining interface code in achieving compatibility between two or more application programs.[130]

---

[126] Machlup, supra note 118, at 281.

[127] Section 14(5): The claim or claims shall –
(a) define the matter for which the applicant seeks protection;
(b) be clear and concise
(c) be supported by the description; and
(d) relate to one invention or to a group of invention which are so linked as to form a single inventive concept.

[128] Bainbridge, supra note 29, at 328.

[129] Robert J. Hart kindly gave this explanation via e-mail contact on Friday 20 Oct 2000 11:33:26 EDT.

[130] See chapter 2 section 3.3.

Therefore, it can be said that allowing computer programs to be patented neither supports the software industry aim of compatibility nor fulfils the "encourage to disclose" theory that is one of the principles of patent law.

3. Reward theory

Under the patent system, a reward for the inventor is the right to use his invention in exclusion of others for a limited period. This right will prevent others from making, using, selling and importing the invention.[131] It is believed that the grant of a monopoly would enable the inventor to make a profit out of his invention by holding off competition.[132] However, the deliberate restraint of competition has a close link with the incentive theory in that the reward is used as an incentive to innovate. This will lead to the development of technology and the growth of industry which will improve the welfare of society. Thus, it can be said that the reward of granting a monopoly has the ultimate objective of serving the public interest.

In considering whether or not the extension of patent law to cover computer programs is appropriate, it has to be considered whether or not the reward will serve the public interest. It has been suggested that there are mainly three ways in which the reward, if given, would be at the detriment of the public interest.[133]

(a) when the control over the new technology is prolonged beyond the brief period contemplated by the law;
(b) when the patents through the way in which they are licensed to competitors are used to regulate competition among them and thereby to restrain competition far beyond the scope of the patent grant; and
(c) when individual companies are allowed to accumulate so many patents that they can control an important part of the technology of the industry..

As for point (a), it is possible in the field of software technology that the incremental improvement of patented computer programs (a mere extension of the first version)

---

[131] Section 60 of the Patents Act 1977.
[132] Meinhardt, supra note 118, at 19; Carl Chan, 'The Patentability of Software Data Structures After *Lowry* and *Warmerdam*' (1998) 32 New England Law Review 899.
[133] Machlup, supra note 118, at 282.

will enable the inventor to obtain a new patent on such an improved computer programs. As successful computer programs can create a "lock-in" effect, the continuation of securing patents for new versions of patented computer programs will ultimately create a permanent monopoly in the market. It is clear that this is not in the public interest. Furthermore, it may be considered that in fact the limited period of 20 years monopoly may not be appropriate for computer programs while it is justified for other kinds of inventions. It is very rare that a specific piece of software is used for 20 years before it is replaced by a newer version.[134] Most commercial software in the market nowadays is replaced with a newer version every 3-5 years.[135] For example, the Windows operating system has introduced a new version in approximately every three years (Windows 95, Windows 98 and Windows 2000); the Microsoft Word application program has also upgraded every 3 years (Word 6, Word 97 and Word 2000). Therefore, it can be inferred from the need of program upgrading that the commercial life of computer programs is much less than 20 years. Thus, it can be said that the 20-year monopoly is an over-reward for investment in the software industry. Also, the inventor of computer programs is usually able to recoup the investment and make profit from having the lead-time in the market. The inventor may not in fact need to rely on patent law for securing an attractive profit.

As for point (b), it is possible that software companies holding patents may use them in a prejudiced way to control competition in some sectors of the software market. However, in principle, such a malpractice is both controlled by the Patents Act 1977 itself[136] and competition law.[137] Thus, it is unlikely that the grant of patents would prejudice the public interest in this way.

As for point (c), however, it is likely that this may happen. This is because there is no limit on the number of patents allowed to be acquired for a single company. Therefore, it is possible that a large company would be able to obtain so many patents that it could control the main part of technology in a given software sector. This

---

[134] Nora M. Tocups and Robert J. O'Connell, 'Patent Protection for Computer Software' (1997) 14 The Computer Lawyer 18.
[135] According to accounting principles, software usually has the commercial life of three years. See, Michael Kavanagh, 'Accounting for dot.coms' (2000) Nov/Dec Accounting & Business 38.
[136] Section 46-54 (Licences of right and compulsory licences), section 70 (remedy for groundless threats of infringement proceedings) and section 71 (declaration or declarator as to non-infringement).

possibility is supported by the fact that the complexity of software technology has increased dramatically over the last decade. To innovate a new product, a company has to invest a substantial capital in R&D. The direct effect in practice is that only large companies, which have a good financial support, will be able to afford the creation of a new software invention. Thus, it will leave room only for large companies to enter the market. This will dry up the possibility of entering the market by SMEs. This would result in a bad economic impact in the EU as most of the European companies are SMEs. The public interest, by the same token, would be prejudiced in that fewer choices of computer programs would be available and this would result in an unnecessary high price. Therefore, allowing computer programs to be patented will not support the reward theory.

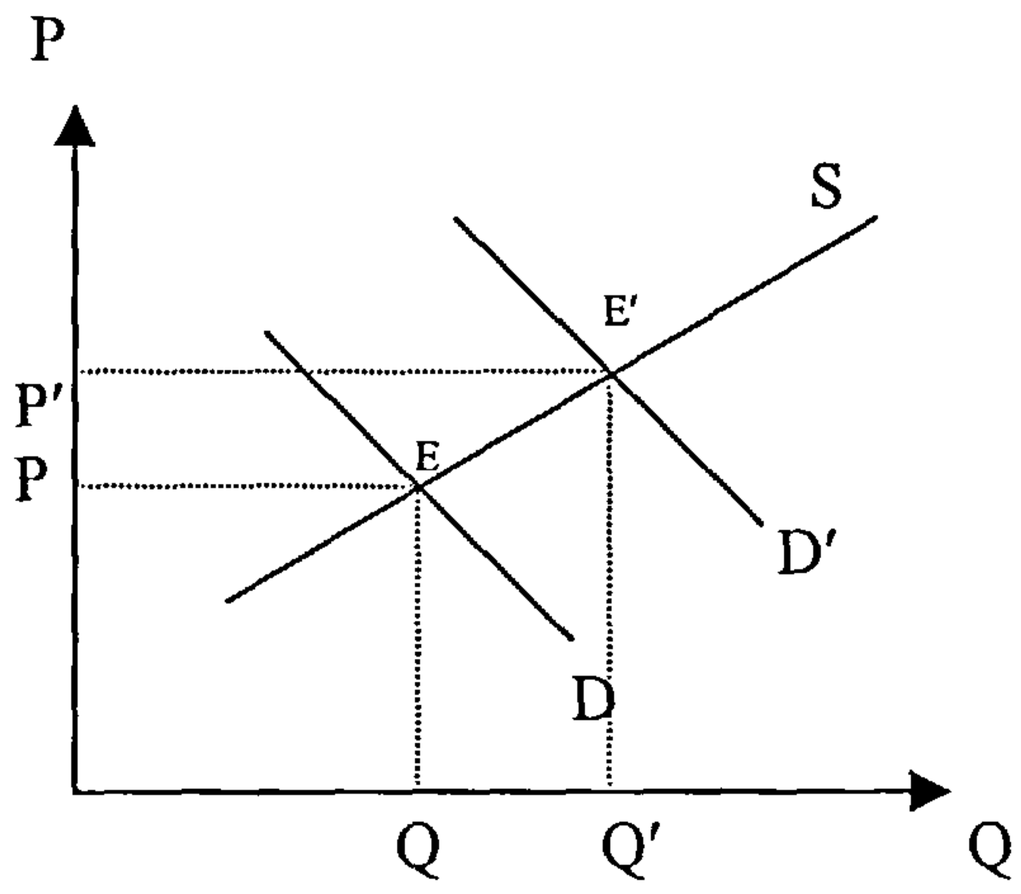
4. Benefit to public's practice theory

It is the aim of patent law that the knowledge is passed to the public after the temporary monopoly has expired so that the public can practise the invention. This aim has to be evaluated when considering the extension of the scope of patent law. A great concern lies in the fact that it can take up to 20 years before the technical knowledge of patented computer programs becomes available to the public. By then, the knowledge is likely to become out-of-date and has little use for the public's practice. This is because the software technology has developed at an unprecedented pace. Within a few years, many new and more powerful programs have been seen in the market. For example, the Internet, which was initially accessible via personal computers (PCs) only, has now been accessible via laptop computers and even mobile phones. Multimedia programs, which a few years ago were able to read only music recorded in the analogue form, have been developed to read and record music recorded in the digital form. Programs, which were developed in the early 80's such as Windows 1.0, now have very little value. As a result, the public are no longer interested in practising or producing them. It may be said that the software technology has developed so rapidly that by the time the patent has expired the public is unlikely to benefit from such an invention. Thus, the "benefit to the public's practice" theory will not be achieved by allowing computer programs to be patented.

---

[137] The Competition Act 1998 and the Treaty of Rome, Article 81 and 82.

Moreover, the grant of patents will unnecessarily raise the price of computer programs. According to the economic theory (Figure 3), patents will result in a monopoly in which the company having a monopoly will be able to make the supply of computer programs constant, i.e. to limit the number of software in the market. In the mean time, the demand will steadily increase as the program will attract more users. Consequently, the demand will be more that the supply. As a result, the supplier (the company having the monopoly) can increase the price without losing customers. This, it is submitted, is not in the public interest.



P axis = Price

Q axis = Quantity

Figure 3

In the normal situation, where there is no monopoly in the market, the demand line (D) meets the supply line (S) at the equilibrium point (E). At this point, the price of the product is (P) and the quantity of the product is (Q). When there is a monopoly in the market, the demand line will shift up as shown by the increasing demand line (D') while the supply line will not change because new suppliers cannot enter into the market. As a result, the price will rise to a new equilibrium point (E′) at a higher price (P′). Therefore, it can be concluded that a monopoly in the market will result in the price increase. This means that a patent, which confers a monopoly, will result in the price increase.

In sum, allowing computer programs to be patented will not support the theories underlying the patent system. This means that the ultimate aim of promoting innovation by using the patent system will not achieved by patenting computer programs.

The justification for the extension of patent law also has to take into account the warning suggested by Professor Cornish. This is to consider what would happen and what would be the extent of infringement if computer programs are to be patented.[138] One of those things that is likely to happen is a "patent war". As expressed by some experts and companies in the software industry, a patent can be used as a negotiating tool when attempting to license or cross-license technology with other companies.[139] Therefore, if computer programs were to be patented, companies would rush into having as many of their programs patented as they could.

There would be so large a number of applications that they would outweigh the number of patent examiners. In the United States, for example, there has been a steep increase in the number of SRI applications, especially for computer-related business methods. The number of applications for computer-related business method has risen from approximately 700 applications per year in 1996 to more than 2,500 applications per year in 1999.[140] As for other patents for SRI, during 1990-1992, there were approximately 5,000 patents granted for SRI.[141] The number granted for SRI is expected to come close to 100,000 by the end of this year.[142] Such a dramatic increase in the number of applications and granted patents has resulted in the need to employ more examiners and it has now been doubted whether or not the examiners all

---

[138] Cornish, supra note 2, at 181.
[139] Tocups and O'Connell, supra note 134, at 19; Oracle Corporation - Patent Policy <http://lpf.ai.mit.edu/Patents/testimony/statements/oracle.statement.html> accessed on 3/12/00; The EuroLinux's petition <http://petition.eurolinux.org/index_html?LANG=en> accessed on 3/12/00.
[140] Seth Shulman, 'Software Patents Tangle the Web' (2000) March/April Technology Review <http://www.techreview.com/articles/ma00/shulman.htm> accessed on 6/12/00.
[141] Gordon Irlam and Ross Williams, *Negative Correlation of Innovation and Software Patents: Revised version of Appendix D of the League for Programming Freedom's submission to the Patent Office, January 25, 1994.* <http://www.base.com/software-patents/scoreboard.html> accessed on 6/12/00.
[142] Seth Shulman, supra note 140.

have the technical capability to evaluate all the patent applications.[143] As a result of this, a search for a prior art and the determination of novelty will become more difficult and complicated. Such unwanted situations are likely to occur if computer programs are allowed to be patented in the EU.

Furthermore, the recent independent study on the economic impact of patentability of computer programs shows that economic efficiency is unlikely to be achieved by having or making SRI patentable.[144] There are four main reasons for this.[145]

(1) There is no evidence that European independent software developers have been unduly affected by the patent positions of large companies or of other software developers.
(2) European independent software developers are making disproportionately little use of the patenting possibilities open to them compared with the use made by large companies and by US SMEs and even independent software developers.
(3) There is increasing but still relatively low use by European independent software developers of patents in raising finance or in licensing i.e. in getting an invention through to being an innovation of benefit e.g. to consumers.
(4) There is considerable evidence of concern by European independent software developers about the potential effects of patents on the development of computer program related inventions.

In addition, the latest UK Government's decision on the issue of whether patents should be granted for software[146] confirms that the exclusion of computer programs from patentable inventions should be retained. The current scope of SRI itself, however, still needs to be clarified.

---

[143] Paul Gosling, 'Who Owns Nature?' (2000) Nov/Dec Accounting & Business 14, 16. There were approximately 104,000 patents issued in 1997 and the number has risen to 289,000 in 1999.
[144] Study Contract ETD/99/B5-3000/E/106: The Economic Impact of Patentability of Computer Programs (Report to the European Commission by Robert Hart, Peter Holmes and John Reid, on behalf of Intellectual Property Institute), <http://europa.eu.int/comm/internal_market/en/intprop/indprop/study.pdf> accessed on 21/10/00.
[145] Ibid., at 3.
[146] The Patent Office, 'Should Patents be Granted for Computer Software or Ways of Doing Business?: The Government's Conclusions' <http://www.patent.gov.uk/about/consultations/conclusions.htm> accessed on 20/3/01.

Therefore, it can be concluded that the ambition of the European Commission and the Administrative Council of the European Patent Organisation to allow SRI to be patented to the fullest extent may not be appropriate. The best solution, it is submitted, is to continue protecting computer programs under copyright law. Although there will be pressure from some large companies to extend the scope of patent law, the Parliament should not allow the patentability of computer programs beyond that allowed under the EPO approach (in the light of the IBM decision) because it will do more harm and good. However, in order to prevent any distorted competition and commercial problems, copyright law should be amended in the way the proposed framework suggests. Therefore, it can be said that the implication of the proposed framework is that it will support this chapter's suggestion not to allow all computer programs to be patentable to the fullest extent. This is because the proposed framework will prevent the use of copyright law to create a quasi monopoly for computer programs that are not qualified for patent protection.

If there is to be a change towards the grant of a temporary monopoly for computer programs at all, it is suggested that a compromise approach should be taken. That is to adopt the utility model protection.

It is submitted that, in light of the inappropriateness of patent law and the uncertainty of the deletion of computer programs from the excluded matter under the EPC, the proposed utility model protection for computer programs is likely to play a major role. This is because the utility model appears to take the middle approach between patent law and copyright law. As discussed previously, the utility model confers a shorter protection and is a swifter means of obtaining a certificate. This seems to be appropriate for computer programs. However, it is submitted that the utility model should require the disclosure of source code in order to avoid the deficiency that occurs to the patent system. Such a compulsory disclosure will facilitate third parties who wish to create an interoperable program.

Nevertheless, it is expected that there will be plenty of computer programs which do not qualify for utility model protection because they cannot provide a "technical character" even though they are of great economic value. For example, word processing programs, accounting/financial programs and programs assisting

mathematical calculations. This means that the source code of these programs is not required to be disclosed. Therefore, the ability to perform reverse engineering under the proposed framework remains essential.

As for the possibility of introducing the utility model in the UK, it has been recognised that there is a strong demand for the harmonisation of such protection. This can be seen from the Green Paper of 19 July 1995[147] and the Report by the Committee on legal Affairs and Citizens' Rights.[148] The said Committee and the European Parliament have already suggested that computer programs be included in the utility model. This suggestion has been accepted by the Commission as can be seen from its amended proposal of 25 July 1999.[149] For this reason, it is likely that the Council will adopt the Directive on the utility model without further ado. Once adopted, the Directive will require the UK to introduce the utility model protection. However, the impact of the utility model protection on the software industry needs to be studied very carefully. Unfortunately, the study of the utility model protection is beyond the scope of this thesis but it is a very interesting area which the author would recommend being explored. The study should evaluate the impact on copyright law, the law of confidence, patent law and competition law, particularly in relation to the right to do reverse engineering.

9. CONCLUSION

From the discussion above, it can be seen that the proposed framework will create an impact on UK patent law in that, as a result of the suggested relaxation of copyright on reverse engineering, there will be pressure towards the broadening of patent protection for computer programs. This impact, in fact, supports the development of UK patent law in moving from a relatively narrow approach to a broader approach established by the EPO in the *IBM* case. That is to say, in the future, the UK court will allow a claim, which is directed to the technical effect that resides in computer programs and to the solution that solves a technical problem. This also means that the

---

[147] COM(95) 370.
[148] A4-0096/99, 25 February 1999.
[149] COM(99) 309.

proposed framework is in line with the UK PO's approach in changing its practice to follow the EPO's decisions.

Moreover, the proposed framework will play an important role if the proposal to amend the EPC is rejected. As illustrated in the analysis of the proposal in this chapter, the author urges that the provisions of Article 52 EPC should be retained since such proposed changes to patent law do not meet the theoretical foundations on which patent law is based. Thus, the change will do more harm than good to the software industry in the European Union. The proposed framework in this situation will come to the rescue of any shortfall (the free flow of information) created by the dominance of copyright protection over computer programs. If another approach is to be introduced, the author suggests that it should be the utility model system because it would, in theory, strike a balance between copyright and patent law. The author strongly recommends that research on this area be carried out as it will make a further substantial contribution to intellectual property and information technology law.

CHAPTER SEVEN

THE ROLE OF COMPETITION LAW

1. INTRODUCTION

In Chapter four, this thesis suggests a new way of solving the misconceived illegitimacy of reverse engineering activities which lead to several commercial problems both in software development and software maintenance. In brief, the thesis suggests that the law should recognise the process of forward engineering and differentiate it from the process of reverse engineering. As a consequence of this differentiation, reverse engineering can automatically be legalised under the general principle of copyright law as it is merely a process of understanding the copyright work and should not infringe any exclusive rights of the holder of copyright in a reverse engineered computer program. It is submitted that this new proposed framework will solve all the commercial problems (raised in Chapter two) which lead

to the distortion of competition in the software market. Indeed, it is this distortion of competition that the function of competition policy also seeks to prevent.

With regard to competition policy, it has long been recognised in the United Kingdom that 'competition lies at the heart of any successful market economy and is crucial to the protection of consumers' interests and the efficient allocation of resources'.[1] This is because competition 'encourages the development of new or improved products or processes and enhances economic growth and living standard'.[2] At international level, a similar competition policy has been adopted by the European Community. This is well summarised in a publication of the Organisation for Economic Co-operation and Development (OECD):

> Competition policy has as its central economic goal the preservation and promotion of the competitive process, a process which encourages efficiency in the production and allocation of goods and services, and over time, through its effects on innovation and adjustment to technological change, a dynamic process of sustained economic growth. In conditions of effective competition, rivals have equal opportunities to compete for business on the basis and quality of their outputs, and resource deployment follows market success in meeting consumers' demand at the lowest possible cost.[3]

It can be seen that the aims of competition policy, both in the UK and in the European Community, have the same goal. This is to ensure that the competition process is not hindered by anti-competitive activity[4] and to prevent the unfair acquisition of market power by individual undertakings without, at the same time, becoming too overprotective of rivals.[5] Observing these aims without a detailed analysis, it seemed that the software market had already been equipped with a legal mechanism that helped guard against any distortion of competition. Therefore, the illegitimacy of any reverse engineering activities which causes the distortion of competition may in theory be remedied by the exercise of competition law. For this reason, one may

---

[1] *The Competition Act 1998: The Major Provisions OFT 400 March 1999* (Office of Fair Trading) p. 2.
[2] Ibid.
[3] *Competition and Trade Policies: Their Interaction* (OECD: Paris, 1984) para. 232.
[4] Supra note 1.
[5] D.G. Goyder, *EC Competition Law 2nd ed.* (Clarendon Press: Oxford, 1993) p. 13.

argue that the proposed framework is in fact redundant since the exercise of competition law will also potentially prevent the distortion of competition.

It is this circumstance that brings about the discussion of this chapter. In this chapter, it will be considered whether or not such an assertion has merit. To what extent competition policy seeks to intervene in undertakings' businesses is still a central issue of controversy among lawyers and economists.[6] This chapter will endeavour to show that competition law is unlikely to achieve the goal when compared to the new proposed framework. Thus, unless the proposed framework is introduced, the current software market is still prone to ineffective competition.

This chapter will begin, as a background for the further analysis, with consideration of the relationship between competition policy and competition in the software reverse engineering market. Then, it will be followed by examination of whether Competition Law can restore competition in the software market which has been distorted as a result of the commercial problems caused by the inability to perform reverse engineering. The most relevant Articles, i.e. Article 81 and 82 of the EC Treaty which are the substance of competition law, will be analysed in separate sections. In section three of this chapter, issues related to Article 81 will be dealt with and those of Article 82 will be scrutinised in section four. Section five will briefly discuss about the "essential facility" doctrine and show why it is not a relevant issue in this chapter. In section six, the issue of nexus, which is one of the obstacles in bringing competition law into play, will be discussed. Finally, in section seven, it will be concluded that the possible use of competition law to legalise reverse engineering is not effective and the proposed framework is, therefore, needed to stimulate competition and the advancement of technology.

2. BACKGROUND : COMPETITION POLICY AND COMPETITION IN THE SOFTWARE REVERSE ENGINEERING MARKET

The appropriate starting point is to explain briefly how competition policy may come into play in relation to the distortion of competition, and who is the party relying on

---

[6] Ibid.

competition policy. This explanation will help provide the scope for the discussion in this chapter.

Normally, the software companies (hereinafter referred to as "the software company") who own copyright in a computer program would seek to prevent third parties, i.e. its rivals and customers, from performing reverse engineering on its computer program. This can be done by not authorising acts restricted by copyright to those third parties (for the purpose of convenience of this chapter's analysis, those third parties are referred to as "the rivals" unless it is indicated otherwise). In this ordinary scenario, the restricted acts normally exercised are the exclusive rights to copy and to make an adaptation of the work. The rivals, who wish to perform reverse engineering but are prohibited from doing so by copyright and contracts, may attempt to seek a permission from competition enforcement authorities such as the Office of Fair Trading or the European Commission. This can be done by making a complaint to these authorities and asserting that the software company has exercised the exclusive rights conferred by copyright in the way that is in breach of competition policy. The rivals may seek, via the complaint to the authorities, to obtain a licence to perform the acts of copying and adapting, which are the crucial process of reverse engineering, for the purpose of software development and/or software maintenance, whichever the rivals' desire may be. Alternatively, the rivals may seek to obtain a compulsory disclosure of the source code of the software company's program.

If the rivals have already illegally performed the act of reverse engineering and been sued by the software company for copyright infringement, the rivals may rely on competition policy and use it as a defence to such copyright infringement. The expected outcome of such a defence is that, if it is successful, the software company will not be able to enforce its copyright and the rivals will be left free to continue their reverse engineering activities. In response, the software company would normally apply for a motion to strike out such a defence. As it is common that the rivals would also make a complaint simultaneously to the enforcement authorities, particularly the European Commission, the court would in this situation stay any competition issues which survive the software company's application to strike out the defence, pending

the outcome of the rivals' complaint to the authorities.[7] This means that the court is usually prepared to determine just the application to strike out the defence but not prepared to continue the proceedings until it receives a decision from the authorities except where the answer to the complaint is clear.[8] The main reason is that it is highly undesirable for the national court and the authorities (in this case, the European Commission) to be deciding the same competition issues at the same time as it may create inconsistency which undermines legal certainty.[9]

It can be seen from the scenarios illustrated above that competition law is normally relied on by the rivals because copyright law confers a strong protection on the software company for its computer program. Although competition law may come into play from two different directions, namely by the rivals' defence against the software company's copyright infringement action and by the rivals' complaint to the authorities, competition law and competition issues before the court are the same as those before the European Commission (or the UK competition authority). This is confirmed by the fact that the latest Competition Act 1998 which came into force on the 1st of March 2000 has mirrored EC competition law. It is evident that Chapter I and Chapter II prohibitions are based on the Treaty of Rome Article 81 and 82[10] respectively. According to the Office of Fair Trading's Guideline, the prohibitions in the Competition Act 1998 apply to business in the same way as the EC competition law, i.e. Article 81 and 82 of the Treaty of Rome.[11] Therefore, agreements and practices which are prohibited under one regime are also prohibited under the other, and those which are permitted would be permitted under both regimes.[12] For this reason, it is adequate and appropriate to discuss and analyse the application of

---

[7] *Philips Electronics N.V. v. Ingman Ltd.*, CH 1997 P. No. 4100-1 <http://www.courtservice.gov.uk/judgments/judg-frame.htm> para. 12-14.
[8] *MTV Europe v. BMG Record (UK) Ltd.* [1997] EuLR 100.
[9] *Philips v. Ingman*, supra note 7, at para. 13.
[10] Formerly Article 85 and 86 respectively. They are renumbered as a result of the Amsterdam Treaty, OJ C340, 10.11.97, p. 143 (see Article 12 of the Amsterdam Treaty and the tables of equivalence referred to in that Article). This does not have any practical impact as it does not determine the reasoning or outcome of the case.
The Treaty of Amsterdam is the result of the Intergovernmental Conference launched at the Turin European Council on 29 March 1996. It was adopted at the Amsterdam European Council on 16 and 17 June 1997 and signed on 2 October 1997 by the Foreign Ministers of the fifteen Member States. It entered into force on 1 May 1999 (the first day of the second month following ratification by the last Member State) after ratification by all the Member States in accordance with their respective constitutional requirements.
[11] Supra note 1, at 3.
[12] Ibid.

competition law to reverse engineering in the light of EC competition law. It may be noted that EC competition law has a direct effect in the national legal orders as it is based on the Treaty of Rome. Direct effect means that the provisions of EC competition law apply to relationships between individuals and create rights directly in respect of the individuals concerned which national courts must safeguard.[13] Section 60 of the Competition Act 1998, indeed, ensures this desirable consistency by providing that the UK authorities must have regard to any relevant decision or statement of the European Commission and the European Court.[14] Although case law existed before the Competition Act 1998 came into force, competition issues concerning intellectual property right aspects argued in the court were based on EC competition law. Thus, the analysis in this chapter will be based mainly on Article 81 and 82 unless it is indicated otherwise.

In principle, to use competition law in order to restrain the enforcement of copyright by the software company, the rivals have to argue that the software company has exercised the exclusive rights in the way that falls foul of either Article 81 or 82 or both of them. Based on this conventional environment, the next part of this chapter begins with the analysis of Article 81 and will be followed by that of Article 82. In such analysis, reference may be made to US antitrust law as the volume of US case law and the depth of analysis on antitrust issues by the US courts have provided persuasive authority. Moreover, the EU-USA Positive Comity Agreement 1998 ensures that competition and consumer welfare between the EU and USA are not impeded by anti-competitive activities, and that co-operative procedures to achieve the most effective and efficient enforcement of competition law are established between the EU and USA. As a result, it is highly likely that the EU and USA will adopt the same approach regarding competition issues. This is to avoid conflicts in competition law enforcement. Thus, anti-competition activities found to be impermissible under US antitrust law are also likely to be prohibited under EC competition law. This means that modern US courts' decisions will provide a highly persuasive authority for the UK courts and competition authorities, although prior

---

[13] Rosa Greaves, "The Herchel Smith Lecture 1998: Article 86 of the E.C. Treaty and Intellectual Property Rights" [1998] EIPR 379, 380.
[14] Supra note 1, at 8-9.

decisions of the US courts might not be consistent with the approach of the European Court.[15]

3. ISSUES RELATED TO AND ANALYSIS OF ARTICLE 81

3.1 Background of Article 81

A short description of Article 81 may be helpful for the analysis in this part. Article 81 is directed at agreements between undertakings. It prohibits collusion that restricts competition and threatens the unity of the common market. Three conditions must be satisfied in order to bring Article 81 into play:[16]

(1) there must be some form of collusion between undertakings,[17]
(2) this collusion must affect trade between member states, and
(3) such collusion has the object or effect of restricting competition within the common market.

If one of the conditions is not met, it is clear that the agreement or conduct of the software company is not caught by this Article. In this part, the main issue is to what extent can the rivals assert that the software company has acted in breach of this Article.

It is submitted that the extent to which the rivals may rely on this Article is reduced significantly by the first condition. This is largely due to the nature of copyright law which entitles the owner of copyright to exploit his or her work to the exclusion of others. Therefore, there is little need for the software company to collaborate with

---

[15] Ian J Lloyd, *Information Technology Law 2nd ed.* (Butterworths: London, Edinburgh, Dublin, 1997) p. 397. For example, the arguments advanced by the defendant in *Lotus v. Paperback* 740 F Supp 37 (1990) and summarily dismissed by the judge would have received at least more measured consideration had they been advanced before the European Court.

[16] Valentine Korah, *An Introductory Guide to EC Competition Law and Practice 6th ed.* (Hart Publishing: Oxford, 1997) p. 41.

[17] "Undertakings" is a broad concept. It covers any collection of resources to carry out economic activities. It embraces a company, partnership, sole trader or an association, whether or not dealing with its members. An inventor was held to be an undertaking when exploiting his invention (*Reuter/BASF* (76/743/EEC) [1976] 2 C.M.L.R. D44, para. 35. See also Korah, Ibid., at 42-44.

others to effect practice that restricts competition. In the event that collaboration does occur, the rivals' chance of success in relying on this Article to defend against copyright infringement depends on the fulfilment of the other two conditions. As for the moment, the focus is on the question why collaboration between software companies to prevent reverse engineering activities does not usually occur.

3.2 Possibility of collaboration between software companies under Article 81

As copyright law confers exclusive rights only on a person or a single undertaking, the owner of copyright usually acts alone in exercising the exclusive rights to prevent third parties from performing reverse engineering on his or her computer programs.[18] Therefore, normally there is no agreement between undertakings that hold copyright in computer programs. Although copyright law may grant exclusive rights to joint authors, the exercise of such exclusive rights must be in the form of an entity. Permission to perform reverse engineering must be obtained from all joint authors; permission from one of joint authors does not prevent the act of reverse engineering from being breach of copyright. In reality, computer programs which have significant commercial value are created by several programmers but those programmers normally work as employees under a contract of employment for a single company. The person who owns copyright is, therefore, that company.[19]

3.3 Copyright licensing

Does this mean that Article 81 has no application at all? Are there other circumstances which the rivals can benefit from this Article? An affirmative answer to these questions can be found from case law. Although the Article requires that there must be some form of collusion between undertakings, this does not necessarily mean that the collusion must be between two or more software companies. The collusion could be an agreement between the software company and the rivals. However, in reality, if the rivals have agreed with the software company the terms of

---

[18] Richard S. Vermut, 'A Synthesis of the Intellectual Property and Antitrust Laws: A Look at Refusals to License Computer Software' (1997) 22 Columbia-VLA Journal of Law & the Arts 27, 32.
[19] Section 11 of the Copyright Designs and Patents Act 1988.

the licence, they would not need to seek help from the court or the competition authorities. Indeed, in such a situation, the rivals would be regarded as having unclean hands if they seek help from the court or competition authorities because they themselves were involved in the illegal agreement in the first place.

However, in practice, the rivals may refuse the licence and assert that the licence is unreasonable, e.g. the software company charges an excessive royalty for the licence. The question that arises is whether this can be a breach of competition law as the rivals have not at this stage entered into contract with the software company: there is, therefore, no relevant agreement between undertakings as required by Article 81. This question was solved in the judgment of Lord Templeman in *British Leyland v. T.I. Silencers*[20] where his Lordship was of the opinion that:

> If, for example, it was proved in evidence that the owners of English copyright were only prepared to grant a licence on terms which created, or helped to create, a breach of Community law, this court, I apprehend, would not grant an injunction against an infringer who desperately needed a licence for his business purposes and was willing to pay a reasonable royalty for the privilege, but was unwilling to accept or assist or acquiesce in any breach of Community law. The Court could award damages in lieu of an injunction based on a reasonable royalty, or the court could accept an undertaking by the defendant to pay a reasonable royalty to be assessed if not agreed. The court would strive to prevent any breach of Community law while, at the same time, preserving for the copyright owner the benefit of and the right to make, in the learned judge's words, 'the ordinary use of their copyright'. But I do not accept that the court would lend countenance to an argument that the ordinary use of copyright included a use which enabled the owner of the copyright to flout European Community law.[21]

This approach was indeed consistent with the decision of the European Commission in *BP Kemi*[22] where it was held that an agreement that had never been signed could be part of an agreement. Thus, it is clear that the scope of Article 81 extends to cover an excessively-charged software licence and other kinds of offensive licences which have not been entered into by the rivals. For example, a licence which limits the

---

[20] *British Leyland v. T.I. Silencers* [1981] CMLR 75 (CA).
[21] Ibid., at 79.

rivals' technical development of the licensed software[23] or a licence which is subject to acceptance by the rivals' supplementary obligations which have no connection with the subject of the main software licence.[24] The latter is commonly known as the "tying" agreement. This type of agreement can be seen in *U.S. v Microsoft*.[25] In this case, the District Court found that Microsoft's conduct of binding Internet Explorer to Windows with contractual and technological shackles in order to ensure the prominent (and ultimately permanent) presence of Internet Explorer on every Windows user's PC system, was an illegal tying agreement. This is because Microsoft could not explain any legitimate business objectives.[26]

However, in reality, although it may be possible for the rivals to prove the allegation of some kinds of offensive licences, it may be extremely difficult to prove that the royalty charged for a licence is excessive.[27] This is because the rivals may not have access to relevant information in respect of pricing which is normally kept secret in the software company. Nonetheless, in practice the software company often considers that by licensing the rivals to reverse engineer its program the software company will expose trade secrets to the rivals and will directly create competitors in both software development and maintenance markets. Normally, the software company chooses to exploit copyright itself as this generates higher profits than income deriving from licensing that copyright.[28] Such copyright licences are granted only where, for some reasons, the software company is unable to exploit the rights itself.[29] As a result, the software company normally chooses not to licence (authorise) the rivals to reverse engineer its program. The software company may also decide not to disclose the source code of its computer program with the aim of reserving the market for itself in order to generate the highest profit possible. Such conduct of the software company will not fall within the ambit of Article 81 since it needs no collaboration with other

---

[22] (79/934/EEC) [1979] 3 C.M.L.R. 684, para. 45.
[23] This kind of licence falls within Article 81(1)(a).
[24] This kind of licence falls within Article 81(1)(e).
[25] *United States of America v Microsoft Corporation*, Civil Action No. 98-1232 (TPJ), <http://usvms.gpo.gov/conclusions_index.html> accessed on 09/6/00.
[26] Ibid., at p 10-14, <http://usvms.gpo.gov/ms-conclusions.pdf> accessed on 09/6/00.
[27] *Philips v. Ingman*, supra note 7, at para. 99. Laddie J. expressed that he had grave doubts as to whether the defendants would be able to prove in the trial that *Philips* charged royalty excessively for the licence to manufacture CD.
[28] Korah, supra note 16, at 256.
[29] Ibid.

companies. In this situation, it begs the question whether refusal to licence will be prohibited by Article 82. This will be examined in section 4.

With regard to the "tying" agreement, normally an undertaking would couple the tying agreement with other commercially offensive conducts such as those in the *Microsoft* case. For example, preventing manufacturers of PCs (known as "original equipment manufacturers" or "OEMs") from removing the ready means of accessing the Internet Explorer and from promoting the Navigator in the boot sequence; and pressuring OEMs to promote the Internet Explorer and to not pre install or promote the Navigator. Therefore, the Court would not normally find breach of competition law by basing its finding on the tying agreement alone. In this situation, it has to be considered whether or not there is a connection between the copyright infringement claim and the defence based on illegal agreements and conducts. This is to be discussed in section 6.

3.4 Effect on trade

For others offensive licences which are plausible to be proved at trial, the rivals still need satisfy the other two conditions, namely the effect on trade and agreements having the object or effect of restricting competition.

Once the rivals can prove that a licence agreement offered by the software company falls within one of the listed examples of the prohibited agreements provided in Article 81(1)(a)-(e), it is relatively easy to fulfil the condition that the collusion must affect trade between member states. This is because the concept of trade has been interpreted very broadly to cover all economic activities relating to goods or services. Given that any influence, direct or indirect, actual or potential, on the pattern of trade between Member States is considered,[30] the European Court has found that even where the parties to an agreement are confined to the same country, inter-state trade may still be affected.[31] Especially, in respect of agreements concerning copyright, it is suggested that it is almost impossible to speculate on a situation where such an

---

[30] See Case 56/65 *Société Technique Minière v. Maschinenbau Ulm GmbH* [1966] E.C.R. 235.

agreement does not affect trade between Member States.[32] Therefore, agreements which affect trade within the United Kingdom will infringe both the Chapter I Prohibition and Article 81.[33] Until the present time, there has been only one case, the *Hugin*[34] case where this condition is not met. Thus, it is fair to conclude that if the other conditions of Article 81 are fulfilled, it would be extremely rare for a case to be dismissed due to the fact that the offensive licence agreement does not affect trade in the UK and between Member States.

3.5 Restriction of competition

The last condition the rivals have to fulfil is that the offensive licence must have the object or effect of restricting competition. Although it is quite clear that an offensive licence, whose characters fall within one of the non-exhaustive list in Article 81(1)(a)-(e),[35] will be considered as having the object or effect of restricting competition, there can be other kinds of offensive licences which do not fall within the list but which may be prohibited under this Article. The difficulties in determining whether those other licences are caught by this Article lie in the fact that, not only the software market but virtually all the product markets encountered within the European Community are imperfectly competitive.[36] Therefore, the task of assessing whether those other offensive licences will have such an object or effect becomes very elusive.

On the positive side, the rivals are eased off by the wording of this condition in that offensive licences may have *either* their object *or* effect on competition. In *Consten*

---

[31] See Case 8/72 *Vereeniging Van Cemethandelaren v. Commission* [1973] C.M.L.R. 7. See also Korah, supra note 16, at 55.
[32] Greaves, supra note 13.
[33] OFT 400, supra note 1, at 16.
[34] [1979] 3 C.M.L.R. 345.
[35] Article 81(1) The following shall be prohibited … and in particular those which:
(a) directly or indirectly fix purchase or selling prices or any other trading conditions;
(b) limit or control production, markets, technical development, or investment;
(c) share markets or sources of supply;
(d) apply dissimilar conditions to equivalent transactions with other trading parties, thereby placing them at a competitive disadvantage;
(e) make the conclusion of contracts subject to acceptance by the other parties of supplementary obligations which, by their nature or according to commercial usage, have no connection with the subject of such contracts.
[36] Goyder, supra note 5, at 116.

and Grundig v. Commission,[37] it was held that there was no need to examine the effects of an agreement if its object was to restrict competition. In the *STM* case,[38] the Court pointed out that the first step was to examine the object of the agreement in question by considering the terms in that agreement. If the agreement did not appear to have such a restrictive object in its economic context, the Court would then move on to consider whether the effect of the agreement resulted in the distortion of competition.[39] This could be answered by comparing the results of the agreement with the likely state of affairs which would exist in the absence of that restriction.[40]

However, the effect on competition caused by the agreement needs to be considered in the context of copyright law which by itself naturally results in a restriction of competition. This is the same question as that which occurs in the context of refusal to license (Article 82) which will be discussed in the next section. Moreover, the validity of asserting breach of Article 81 as a defence to copyright infringement has to be relied on the issue of nexus. This will be discussed in section 6.

From the software company's legal standpoint, it may be considered that it is more risky to offer a potentially illegal licence than not to offer the licence at all because the former to be found in breach of competition law is not required to abuse its dominant position. In contrast, by not offering the licence, the software company will have a chance to argue that it does not hold a dominant position. In addition, such refusal to license cannot automatically be considered as an abuse of a dominant position as will be seen in the analysis in the next part. It is also argued that the main reason why the software company normally takes the refusal to license approach is to maximise the profit.

4. ISSUES RELATED TO AND ANALYSIS OF ARTICLE 82

As discussed in the previous section, the software company may refuse to authorise the rivals to perform the acts of copying and adaptation which form the major part of

---

[37] (56&58/64) [1966] E.C.R. 299.
[38] *Société Technique Minière v. Maschinenbau Ulm GmbH*, supra note 30.
[39] Ibid., at 249.
[40] Ibid., at 250.

reverse engineering activities. This begs the question whether this practice can be caught by competition law, especially Article 82.

4.1 Two kinds of refusal

Before moving on to the analysis on the question of law, regard must be paid to the forms in which a computer program is expressed. So far as reverse engineering is concerned, the software company may refuse to authorise the rivals to perform the acts of copying and adaptation on the computer program expressed in the form of object code. It may be noted that, normally, computer programs are acquired by means of licensing rather than one-off purchasing. The terms of the licence also normally authorise the buyer or licensee to perform the acts of copying and adaptation only insofar as it is necessary for a normal use of the program. Therefore, although the buyer or licensee acquires the object code of the program, he or she cannot use it for the purpose of reverse engineering (keeping in mind that the discussion is based on the hypothesis that the Software Directive and the CDPA 1988 are deficient for permitting reverse engineering). Hence, it may be stated, albeit technically incorrect, that the software company may refuse to license the rivals to perform reverse engineering.

The other type of refusal to license is related to the source code of a computer program. This type of refusal may be said to come from the situation where the rivals cannot perform reverse engineering on the program they acquire in the form of object code, owing to the terms of the licence. An alternative solution for the rivals is to seek to obtain a copy of the original source code of the program from the software company. In this circumstance, the software company may refuse to license the source code of the program to the rivals. It may be noted that for this type of refusal, the issue of reverse engineering may not be directly involved. However, since this type of refusal stems from or is related to the first type of refusal, it is necessary to analyse its legal position since the result of the analysis will give an answer to the question of how this thesis' proposed framework will affect competition law.

4.2 Brief background of law

Before beginning the analysis of refusal to licence, a brief description of Article 82 may be helpful. Generally, Article 82 is not expressed to prohibit anti-competitive structures or conduct that leads to them, i.e. the existence or acquisition of market power, but it aims to restrain conduct by a dominant firm that harms those with whom it deals.[41] The main elements of Article 82 are that:

(1) the undertaking must be in a dominant position;
(2) the undertaking abuses its dominant position; and
(3) such an abuse affects trade between Member States.

As the third element has been construed similarly to that of Article 81[42] (i.e. it has been interpreted so broadly that even though the abusive conduct or practice of the undertaking is confined to a Member State, trade in the common market is also likely to be affected[43]), it is likely that if the first two elements are fulfilled, the conduct of the undertaking will be prohibited. Thus, the analysis in this part will focus on the first two elements.

4.3 Analysis of the first element

4.3.1 Ambiguity in the concept

According to the first element, the rivals need to prove that the software company is in a dominant position. A dominant position of an undertaking, however, is not easy to determined. As will be illustrated in the analysis below, the Commission and the European Court have failed to give a clear view of what would constitute a dominant position.

---

[41] Korah, supra note 16, at 77.
[42] Ibid., at 61. See also *Commercial Solvents v. Commission* (Case 6&7/73) [1974] E.C.R. 223, para. 32.
[43] See discussion in section 3.4 of this chapter.

The ambiguity is rooted in the concept of a dominant position itself, chiefly because of the fact that there is no statutory definition of market dominance.[44] In the early 1970s, the concept seemed to embrace an economic sense. This concept was defined by the Commission in *Continental Can*,[45] whose opinion was endorsed by the Advocate General and impliedly accepted by the European Court,[46] as follows:

> Undertakings are in a dominant position when they have the power to behave independently, which puts them in a position to act without taking into account their competitors, purchasers or suppliers. That is the position when, because of their share of the market, or of their share of the market combined with the availability of technical knowledge, raw materials or capital, they have the power to determine prices or to control production or distribution for a significant part of the products in question. This power does not necessarily have to derive from an absolute domination permitting the undertakings which hold it to eliminate all will on the part of their economic partners, but it is enough that they be strong enough as a whole to ensure to those undertakings an overall independence of behaviour, even if there are differences in intensity in their influence on the different partial markets.[47]

The focus of this concept is on the discretionary power of the undertaking to set prices or to control production or distribution of the product without having competitive pressure from other parties. The problem with this concept is that it places too much emphasis on economic relationship between the undertaking and its competitors, thereby ignoring the effect on the public in the situation where competition remains unfettered.[48]

---

[44] Cini and McGowan state that the omission is quite understandable when we realise that many economists accept that economic theory offers little guidance at to what is meant by a dominant position of a kind which could be of substantial help in framing a legal definition. See Michelle Cini and Lee McGowan, *Competition Policy in the European Union* (Macmillian Press Ltd: Basingstoke, 1998).

[45] *Europemballage Corporation and Continental Can Company Inc. v. Commission* (Case 6/72) [1972] C.M.L.R. D 11, [1973] E.C.R. 215.

[46] Korah, supra note 16, at 78.

[47] Supra note 45, at para II. 3.

[48] Goyder, supra note 5, at 356.

However, the focus has been shifted away from the economic concept to a purely legal concept. This legal concept was first introduced in the *United Brands* case.[49] In this case, the Court defined the concept as follows:

> The dominant position referred to in this Article relates to a position of economic strength enjoyed by an undertaking which enables it to prevent effective competition being maintained on the relevant market by giving it the power to behave to an appreciable extent independently of its competitors, customers and ultimately of consumers.[50]

Although this concept takes into account the power of an undertaking to behave independently which seems to reflect also the economic concept of power over pricing, the overall effect of this legal concept shows a different outcome.[51] This was clearly illustrated in the *United Brands* case. In this case, the *United Brands* was found in a dominant position because it could prevent dealers from buying the bananas at the ports of entry and selling them on, although it could not set the price independently of its major competitor and made losses in four out of the last five years.[52] Although this concept has been followed, as will be discussed below, in *Hoffman La Roche*,[53] it does not fit comfortably with a position of economic strength enjoyed by the software company. Since developing and maintaining commercially valuable software involve a considerable amount of investment, it can be said that the economic strength of the software company depends on the ability to set the price above the point at which no returns would be gained. Also because the software industry is a fast-moving one, the lack of the company's power over pricing would be severely detrimental to its business as there will be insufficient revenue for developing the next generation of the software which would help the company remain competitive. A good example of this situation can be seen from the fading away from a dominant position of Netscape company[54] in the Internet browser market, as a result of Microsoft's[55] marketing tactics (giving away the Internet Explorer for free) which

---

[49] *United Brands Co. (New Jersey, U.S.A.) and United Brands Continental B.V. (Rotterdam) v. Commission* (Case 27/76) [1978] E.C.R. 207.
[50] Ibid., at para. 65.
[51] Korah, supra note 16, at 78.
[52] *United Brands*, supra note 49, at para 125-129. See also Korah, Ibid., at 79.
[53] Infra note 57.
[54] Netscape Communications Corporation.
[55] Microsoft Corporation.

abate Netscape's power over pricing.[56] Therefore, unlike the banana market in the *United Brands* case, if the software company cannot control the price, it would be unlikely that the company would enjoy economic strength that suggests a dominant position.

Apart from uncertainty of the definition and its potential inapplicability to the software field discussed above, the European Court's attempt to clarify the definition in a subsequent case, *Hoffman La Roche*,[57] did not prove to be successful but made the matter more complicated. The European Court went on to give further explanation as follows:

> Such a position does not preclude some competition, which it does where there is a monopoly or a quasi monopoly, but enables the undertaking which profits by it, if not to determine, at least to have an appreciable influence on the conditions under which that competition will develop, and in any case to act largely in disregard of it so long as such conduct does not operate to its detriment.[58]

According to this explanation, a dominant position may be found at the level well below monopolies. A question arises as to whether or not such a wide concept of a dominant position is appropriate for determining a dominant position of the owner of copyrighted products, especially computer programs. As copyright law confers monopoly, by way of exclusive rights, on the owner of copyright to a certain extent, it can be logically inferred that the owner of copyright would automatically be held in a dominant position. This may seem inappropriate in reality, for in many circumstances the owner of copyright may not secure a large share in the market. How can this concept of a dominant position apply to markets of copyrighted products? It can be said that the answer is found in the European Court's decision in *Deutsche Grammophon*.[59] The Court stated that:

---

[56] Netscape used to have 80 percent market share in the Internet browser market. See *U.S. v. Microsoft* (findings of fact) para. 166, at <http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm>.
[57] *Hoffman La Roche v. Commission* (Case 102/77) [1979] 3 C.M.L.R. 211; <http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexplus!prod!CELEXnumdoc&lg=en&numdoc=61976J0085> accessed on 01/08/00.
[58] Ibid., at para 4 (of the Internet reference).
[59] *Deutsche Grammophon Gesellschaft mbH v. Metro - SB Grossmarkte GmbH &Co. KG* (Case 78/80) [1971] E.C.R. 487.

The manufacturer of sound recordings who holds a right related to copyright does not occupy a dominant position within the meaning of Article [82] of the Treaty merely by exercising his exclusive rights to distribute the protected article. Article [82] further requires that the manufacturer should have the power to impede the maintenance of effective competition over a considerable part of the relevant market – in particular to the existence of any producers making similar products and to their position on the market.[60]

The Court's suggestion was that the emphasis should be placed on the real market power of the owner of copyright in exercising exclusive rights to restrict competition in the relevant market, not on the exclusive rights *per se*. This approach was adopted in subsequent cases such as in *Volvo v Veng*.[61] In *Volvo v Veng*, the Court stated that 'it is apparent from previous decisions of the Court that the mere possession of an industrial property right does not automatically imply that the holder thereof occupies a dominant position within the meaning of Article [82]'.[62] Therefore, it can be seen that the rivals will face an initial hurdle in showing whether the software company holds a dominant position and why the software company's possession of copyright occupies a dominant position.

4.3.2 Ambiguity in the assessment of the existence of a dominant position

i. The product market

Difficulties for the rivals do not only lie in the concept of a dominant position itself, but also in assessing the existence of a dominant position. Dominance can only be assessed by reference to a defined category of products within a specified geographical area.[63] The narrower the definitions of the product market and the geographical market, the easier it is to conclude that an undertaking acquires a

---

[60] Ibid., at para. 16.
[61] *Volvo AB v. Erik Veng (U.K.) Ltd* (Case 238/87) [1989] 4 C.M.L.R. 122.
[62] Ibid., at 128.
[63] Jonathan Faull and Ali Nikpay, *Fuall & Nikpay: The EC Law of Competition* (Oxford University Press: Oxford, 1999) p. 125.

dominant position.[64] The question arising here is how to determine the scope of the product market and the geographical market.

With respect to the product market, the general approach of the European Commission and the European Court has been to focus on "interchangeability": the extent to which the goods under scrutiny are interchangeable with other products.[65] This in turn raises the question of how "interchangeability" can be defined. On the one hand, if "interchangeability" includes substitutes that are not perfect, the scope of the product market will be very wide, resulting in an understated market share of an undertaking. On the other hand, if "interchangeability" is restricted to only perfect substitutes, the product market will be very narrow. This can exaggerate the market power of the undertaking. To draw the line between these two extremes is not an easy task. Until now, the Court does not seem to have adopted a consistent approach. Moreover, each formula that the Court creates does not prove to be perfect and often appears to apply uncomfortably to another case. This will be discussed below.

In principle, interchangeability of a product has to be measured from both the demand and supply sides of the market.[66] This approach was adopted in *Continental Can*[67] and was welcomed by many economists.[68] The Court in *Continental Can* disagreed with the approach of the Commission which concentrated only on the demand side of the market. The Commission considered the demand for cans used for meat and fish products and the possibility of using other kinds of containers such as plastic and glass containers instead of cans. However, the Commission did not consider substitutes on the supply side. The Court pointed out that by not considering both sides of the market, a dominant position of an undertaking could be misconceived. In this case, the Court stated that:

> a dominant position on the market for light metal containers for meat and fish [could] not be decisive, as long as it [had] not been proved that competitors

---

[64] Paul Craig and Grainne de Burca, *EC Law: Text, cases, & Materials* (Clarendon Press: Oxford, 1995) p. 941; Guy Tritton, *Intellectual Property in Europe* (Sweet & Maxwell: London, 1996) p. 612.
[65] Craig, ibid. The concept of interchangeability has other economic labels, namely cross-elasticity of demand and demand substitutability. All three terms have been used by the Court and Commission and essentially mean the same. See Tritton, ibid., at 613.
[66] Craig, ibid.
[67] Supra at note 45.

> from other sectors of the market for light metal containers [were] not in a position to enter this market, by a simple adaptation, with sufficient strength to create a serious counterweight.[69]

However, the approach of the Court was not flawless even though it considered both the demand and supply sides of the market. The Court did not consider the possibility of a completely new entrant who might obtain a technology licence from another established company, for example, American Can.[70] The Court simply looked at the possibility of entering the market by the maker of cylindrical cans who might be interested in entering the market by making the irregular shaped cans used for meat and fish as alternatives from those made by Continental Can. This omission may result in unnecessarily stronger market power of the undertaking which could lead to an incorrect finding of a dominant position. Despite this imperfect consideration, the approach of the Court in considering both sides of the market seems to be appropriate for the consideration of a dominant position in the context of the computer software market. Since computer technology has developed so rapidly, the interchangeability on the supply side has had a great effect on the consumer's choice in that it has widened the product market of the software in question, which would in turn affect the finding of a dominant position of the software company.

However, as stated above, the European Court did not retain a consistent approach. It changed the approach to focus on the demand side of the market, namely that only substitutes on the demand side were taken into account. This was clearly illustrated in *United Brands* where the Court defined the relevant product market by merely looking at substitutability of bananas. The Court stated that:

> 22 For the banana to be regarded as forming a market which is sufficiently differentiated from other fruit markets it must be possible for it to be singled out by such special features distinguishing it from other fruits that it is only to a limited extent interchangeable with them and is only exposed to their competition in a way that is hardly perceptible.
>
> ...

---

[68] Korah, supra note 16, at 80.
[69] Supra at note 45, at para. 33.
[70] Korah, supra note 16, at 80.

> 27 Since the banana is a fruit which is always available in sufficient quantities the question whether it can be replaced by other fruits must be determined over the whole of the year for the purpose of ascertaining the degree of competition between it and other fruit.
>
> 28 The studies of the banana market on the court's file show that on the latter market there is no significant long term cross-elasticity any more than – as has been mentioned – there is any seasonal substitutability in general between the banana and all the seasonal fruits, as this only exists between the banana and two fruits (peaches and table grapes) in one of the countries (West Germany) of the relevant geographic market.
>
> ...
>
> 34 It follows from all these considerations that a very large number of consumers having a constant need for bananas are not noticeably or even appreciably enticed away from the consumption of this product by the arrival of other fresh fruit on the market and that even the personal peak periods only affect it for a limited period of time and to a very limited extent from the point of view of substitutability.
>
> 35 Consequently the banana market is a market which is sufficiently distinct from the other fresh fruit markets.[71]

It can be seen that this approach would result in a very narrow market. If this were to apply to the field of computer software, two inappropriate outcomes could be expected. The first is that the software company (the undertaking) will always be in a dominant position if the product of the company is a program which has unique features or which is designed for a particular purpose (bespoke software). For example, a program for calculating the account of a company or creating databases for an organisation. In this circumstance, substitutes of these programs will not be commonly available from the point of view of the demand side. However, the profitability of such a program may attract other companies to enter the market by offering the customer the creation of a similar program. This will reduce the market power of the first software company significantly, even though the software company seems to occupy a dominant position (where the interchangeability on the demand side is considered only). Therefore, a misinterpretation of a dominant position of the software company may occur if the *United Brands* approach is applied.

---

[71] Supra note 49, at para 22, 27, 28, 34 and 35.

The second outcome is that the software company would also readily be found in a dominant position in the maintenance market of its products. This is because there is virtually no substitute on the demand side for the computer program in question. In other words, the maintenance service of other computer programs will not be appropriate for the computer program in question: software tools or diagnostic programs designed for one program may not be used for another. Therefore, the customer or the user will not be able to switch to maintenance services offered for other programs, but be confined to the maintenance service offered by the software company for the computer program in question. Thus, it can be seen that the approach established in *United Brands* does not fit comfortably in the field of computer software as the software company would always be found in a dominant position. This finding will easily be negated if the supply side has been taken into account.

The inconsistency of the European Court's approach has not been limited to the two contrasting approaches that it adopted in its decisions in *Continental Can* and *United Brands* discussed above. There is another approach which appears to emerge from *Michelin*.[72] In this case, the Court reverted to the approach it adopted in *Continental Can* – that is to consider the interchangeability both on the demand and supply sides of the market. However, instead of relying on the idea of freedom from competitive constraints as in *Continental Can*, it relied on the idea of an ability to impede competition.[73]

In *Michelin*, the Court considered the structure of demand and supply on the market and found that, on the demand side, there was no interchangeability between car and van tyres on the one hand and heavy-vehicle tyres on the other hand.[74] For the supply side, there was no elasticity of supply between tyres for heavy vehicles and car tyres either, due to significant differences in production techniques and tools needed for such manufacture.[75] In addition, the Court found that there were other criteria and

---

[72] *NV Nederlandsche Banden Industrie Michelin v. Commission* (Case 322/81) [1983] E.C.R. 3461.
[73] See and compare *Continental Can* – supra note 45, at para. 24, 25 and 28 – and *Michelin* – ibid., at para. 30 and 37. See also Korah, supra note 16, at 80.
[74] Supra note 72, at para. 37 and 39.
[75] Supra note 72, at para. 41. The Court found that the fact that time and considerable investment were required in order to modify production plant for the manufacture of light-vehicle tyres instead of heavy-vehicle tyres or vice versa meant that there was no discernible relationship between the two

evidence supporting the market power of Michelin. For instance, Michelin's network of commercial representatives gave it direct access to tyre users at all times and enabled it to maintain and strengthen its position on the market and to protect itself more effectively against competition. Therefore, the Court held that Michelin occupied a dominant position.[76]

The Court's approach in the *Michelin* case appears to fit comfortably in the computer software context as the interchangeability of the product must be measured from both the demand and supply sides of the market, otherwise the determination of a dominant position will be misleading as described above. In addition to its consideration of interchangeability on both sides of the market, the Court considered the particular fact of the case which gave Michelin extra market power to prevent effective competition, namely its network of commercial representatives. This sets the trend that a special circumstance may play an important part in determining a dominant position of an undertaking. Consistently, the Court in the *Magill* case[77] has followed this trend. The Court also developed the principle further to the point that, in determining a dominant position, it relied solely on a special circumstance of this case, viz. by force of circumstance, the broadcaster companies enjoyed a *de facto* monopoly over the information used to compile listings for the TV programmes.[78]

Although the Court has developed its approach to determine a dominant position on a case by case basis, it still retains the main principle that interchangeability on both sides of the market has to be considered. This can be seen from its decision in *Tetra Pak*,[79] where the Court cited its decision in *Michelin* and confirmed that 'the competitive conditions and the structure of supply and demand on the market [were] relevant criteria for determining whether certain products [were] interchangeable with others',[80] Therefore, it can be concluded that there are four approaches which the

---

categories of tyre enabling production to be adapted to demand on the market. This was the reason why 1n 1977, when the supply of tyres of heavy vehicles was insufficient, Michelin NV decided to grant an extra bonus instead of using surplus production capacity for car tyres to meet demand.

[76] Supra note 72, at para 58 and 60.

[77] *Radio Telefis Eireann (RTE) and Independent Television Publications (ITP) v. Commission* (Case C-241-42/91 P.) [1995] E.C.R. I-743, [1995] 4 C.M.L.R. 718. This case is commonly known as the *Magill* case.

[78] Ibid., at para. 47.

[79] *Tetra Pak International SA v. Commission* (Case C-333/94 P.) [1996] E.C.R. I-5951.

[80] Ibid., at para. 13.

Court uses to determine a dominant position. It is submitted that the variety and inconsistency of the Court's approaches will cause legal uncertainty which will create difficulties for the rivals to self-determine whether or not the software company they complain is in a dominant position.

ii. The geographical market

With respect to the geographical market, the European Court generally seems to adopt a narrow approach, unless it is clear that the product is sold worldwide.[81] Nevertheless, so far as reverse engineering is concerned, it seems that the analysis of the geographical market is not relevant since the information the rivals seek to obtain by way of reverse engineering is not available from anywhere else but the software company. Therefore, the software company will always in a dominant position, regardless of whether or not the geographical market is limited to one Member State, the common market or the global market.

4.4 Analysis of the second element

4.4.1 Uncertainty of judging abuse of a dominant position

Apart from the legal uncertainty found in the determination of a dominant position, the other major difficulty for the rivals in relying on Article 82 for their defence to copyright infringement lies in the prove of abuse of a dominant position (the second element). It has been unequivocally recognised since *Parke, Davis & Co.*[82] that although intellectual property rights (in this case it was patent) confer on the holder a special protection, it does not follow that the exercise of the exclusive rights thus conferred implies the fulfilment of the elements of Article 82, unless the exercise of the exclusive rights has degenerated into an abuse of the abovementioned protection.[83]

This principle is endorsed in *Volvo v. Veng* where the Court stated that:

---

[81] See *Wood Pulp* [1993] E.C.R. 1307, para. 12-13.
[82] *Parke, Davis & Co. v. Probel, Centrafarm and others* (Case 24/67) [1968] E.C.R. 55.
[83] Ibid., at 72.

> The mere acquisition of an industrial or commercial property right (and the exercise of the corresponding rights without which registration of the design would be deprived of any practical utility) does not therefore constitute abuse of a dominant position. A further element is required.[84]

This well-known principle produces an initial obstacle for the rivals in bringing the conduct of the software company within the ambit of competition law. This is because they will not be able simply to argue that the exercise of exclusive rights by the software company constitutes an abuse of a dominant position. The rivals need to show that the software company's use of the exclusive rights has degenerated into an abuse of the legal protection conferred by copyright.

For the software company to prevent the act of reverse engineering being carried out by the rivals, it may simply exercise its exclusive rights by not authorising the acts of copying and making an adaptation of the object code of its program. The question is that, is such refusal to authorise copying and adapting regarded as degenerating into an abuse of the copyright protection? On the one hand, it may be argued that such refusal aims to impede competition and create or maintain the monopoly in the market. Therefore, the exercise of the exclusive rights has degenerated into an abuse of the protection provided by copyright. Although this argument is quite convincing, the authority deriving from case law suggests the opposite. The High Court, in a copyright case of *Yale Security Products v. Newman*,[85] held that 'since English law [gave] a copyright owner certain monopoly rights and since any assertion of a monopoly necessarily limit[ed] competition, the limitation that naturally result[ed] from enforcing the statutory monopoly [was] not in itself an abuse'.[86] The High Court stated further that 'to constitute an abuse, there must be special circumstances making the enforcement of monopoly contrary to the Rome Treaty'.[87] However, the High Court left unsaid what could be considered as special circumstances. It is submitted that this approach may create legal uncertainty in the future in that this approach will influence subsequent courts to determine the issue on a case-by-case basis. So far, there is only one case which the European Court did find special

---

[84] Supra note 61, at 130.
[85] *Yale Security Products Ltd. v. Newman* [1990] FSR 320.
[86] Ibid., at 324.
[87] Ibid.

circumstances. This is the *Magill* case.[88] The importance and application of this case will be discussed below as the discussion on "refusal to license" will also lead to this case.

4.4.2 Refusal to license

In addition to the exercise of the exclusive rights by not authorising the acts of copying and adapting, the software company is also able to refuse to disclose to the rivals the source code of its computer program. This kind of refusal can be considered as the exercise of inherent right of the exclusive rights provided in every kind of intellectual property right, which in essence gives value to intellectual property works.[89]

As mentioned earlier, software is normally acquired by way of licensing. Thus, it can also be viewed that the exercise of the exclusive rights by not authorising the acts of copying and adapting amounts to refusal to licence the rivals to perform such acts. This can be regarded as the other kind of refusal to license. However, the ultimate purpose of both kinds of refusal is to prevent access to information contained in the source code of computer programs. Therefore, the analysis below will not distinguish between these two kinds of refusal. The issue for the analysis of this part is whether the rivals can assert that such refusal constitutes an abuse of a dominant position.

Once again, the rivals are facing an immediate obstacle from the principle established in *Volvo v. Veng* that refusal to license cannot in itself constitute an abuse of a dominant position. The European Court was of the opinion as cited in the following passage.

> It must also be emphasised that the right of the proprietor of a protected design to prevent third parties from manufacturing and selling or importing, without its consent, products incorporating the design constitutes the very subject-matter of his exclusive right. It follows that an obligation imposed

---

[88] *Radio Telefis Eireann (RTE) and Independent Television Publications Ltd (ITP) v. Commission*, supra note 77.

[89] It can be viewed that in fact the exercise of the inherent right gives rise to the need for reverse engineering of the object code of computer programs.

upon the proprietor of a protected design to grant to third parties, even in return for a reasonable royalty, a licence for the supply of products incorporating the design would lead to the proprietor thereof being deprived of the substance of his exclusive right, and that a refusal to grant such a licence cannot in itself constitute an abuse of a dominant position.[90]

The Court thus concluded that:

> Article [82] of the EEC Treaty must be interpreted as meaning that refusal by the proprietor of a registered design to grant licences which would enable third parties to supply body panels covered by that design in return for the payment of a reasonable royalty does not of itself constitute abuse of a dominant position, since that refusal is no more than the consequence of the exercise of the right associated with the registered design.[91]

However, the Court left some possibilities for finding an abuse of a dominant position as it made an exception that if the exercise of exclusive rights involved certain abusive conduct, such an exercise might be prohibited. The Court stated as follows.

> It must however be noted that the exercise of an exclusive right by the proprietor of a registered design in respect of car body panels may be prohibited by Article [82] if it involves, on the part of an undertaking holding a dominant position, certain abusive conduct such as the arbitrary refusal to supply spare parts to independent repairers, the fixing of prices for spare parts at an unfair level or a decision no longer to produce spare parts for a particular model even though many cars of that model are still in circulation, provided that such conduct is liable to affect trade between member-States.[92]

This approach suggests the trend that, in intellectual property-related cases, some special circumstances must be present in order to bring refusal to license within the ambit of Article 82. Although the Court gives a guideline by providing examples of certain abusive conduct, it is inevitable that this will lead to legal uncertainty because the guideline by way of examples is by no means exhaustive. Thus, it is open to the

---

[90] Supra note 61, at 135 (para. 8).
[91] Supra note 61, at 133.
[92] Supra note 61, at 135-36 (para. 9).

courts in subsequent cases to determine the issue on the ground of the facts of each case.

Until the present time, only in the *Magill* case has the European Court found that there was a special circumstance which made refusal to license an abuse of a dominant position. In this case, the Court identified four factors which constituted a special circumstance. Firstly, the appellant broadcasters were the only sources of the basic information on programme scheduling which was the indispensable raw material for compiling a weekly television guide.[93] Secondly, the appellants' refusal to provide basic information prevented the appearance of a new product which the appellants did not offer and for which there was a potential consumer demand.[94] Thirdly, there was no justification for such refusal.[95] Finally, the appellants reserved to themselves the secondary market of weekly television guides by excluding all competition on that market.[96] A question flowing from the Court's approach in the *Magill* case is how this will affect and apply to the rivals' reliance on competition law. Can the rivals argue that, as a result of the *Magill* case, the software company's refusal to license constitutes an abuse of a dominant position? To answer this question, regard must be paid, by way of analogous analysis, to the factors which the Court considered as constituting a special circumstance, thereby making the exercise of exclusive rights of an undertaking coming under the review of Article 82.

i. Software development

In relation to software development, it is rather difficult to argue that the software company is the only source of information for developing a competing product or a non-competing and non-interoperable product. There may be a number of computer programs in the market which perform similar functions to the computer program of the software company. Thus, similar information could also be obtained from those other programs. The decision of the rivals to choose the program of one software company does not preclude the possibility of obtaining similar information from other companies producing similar programs. However, if the rivals wish to create an

---

[93] Supra note 77, at para. 53.
[94] Supra note 77, at para. 54.
[95] Supra note 77, at para. 55.

interoperable product, the software company may be the sole source of information regarding interface information which is necessary for interoperability between two programs. Therefore, it can be seen that only when the rivals have the aim of developing interoperable products does the refusal by the software company come within the first factor.

It is also controversial for the second factor too. In the *Magill* case, it was unambiguous that the refusal did prevent the emergence of a new product but in the case of computer software the position may be different. Refusal to allow the rivals to access ideas and principles of a computer program is not an absolute barrier to the creation of a new program, although it may cause the rivals to spend more time, effort and expenditure in so creating one. Even though it may be argued that such a refusal prevents the emergence of a new interoperable program, this argument may not be sustained in the light of the Java technology since computer programs developed by using this technology do not need to rely on interface information solely available from the software company. Hence, it seems that only the rivals' wish to create an interoperable program based on conventional technology will make the software company's refusal to license analogous to the second factor in the *Magill* case.

For the question of justification for refusal, the third factor, the Court held that there was no justification for the broadcasters' refusal either in the activity of television broadcasting or in that of publishing TV magazines.[97] The Court based its decision on the fact that the broadcasters' refusal was arbitrary because it was not justified by the requirements peculiar to the activity of publishing TV magazines. The refusal made it possible for the broadcasters to adapt to the conditions of a TV magazine market, which was open to competition, in order to ensure the commercial viability of their weekly publications.[98] Similarly, the same reasoning can be applied to refusal to license in the software context. The rivals can argue that the software company's refusal would be arbitrary and unjustified on the ground that the software company would be able to impede unnecessarily competition in the related market. However, in the *Magill* case the broadcasters did not provide "justification" for their refusal

---

[96] Supra note 77, at para. 56.
[97] Supra note 77, at para. 55.
[98] Supra note 77, at para. 30.

because they considered that the refusal fell within the specific subject-mater of the copyright which did not require them to justify their behaviour. By learning from the *Magill* case, the software company may attempt to provide "justification" for its refusal. The software company may argue that its refusal is justified because, for example, the information is protected by the law of confidence as being trade secrets. If this is the case, it is likely that the software company's refusal to license will not constitute an abuse of a dominant position.

For the last factor, the rivals have to argue that the software company's refusal to license would result in a reservation of the market for itself by excluding all competition in that market. In the TV listings magazine market, all competition is excluded simply because to enter into this market a company such as Magill has to obtain the information protected by copyright from the broadcasters. However, in the software market, the software company's market may exclude part but not all of the competition in that market. To enter into the market of the software company, the rivals do not always require information from the software company. The rivals may create a new program from scratch. However, for the interoperable product market, the rivals' ability to enter the market may depend heavily on access to the interface information controlled by the software company. Without such access, it may be extremely difficult for the rivals to create an interoperable program. With the advent of the Java technology as mentioned above, an interoperable program may not, anymore, require access to the interface information. Therefore, it means that not all competition in that market is excluded. For this reason, it is submitted that the rivals are unlikely to argue that the software company's refusal to license constitutes an abuse of a dominant position.

ii. Software maintenance

For software maintenance, the rivals seem to have a better chance to argue that the software company's refusal to license constitutes a similar special circumstance, thereby being in breach of competition law. It is indisputable that the software company is the only source of information necessary for maintaining a computer program it designs or creates. To maintain a computer program efficiently, one needs

to know the detailed information of how that computer program works, i.e. to know source code of the program. The information obtained from the source code will be used to create a diagnostic program for maintaining the main program. Without access to source code, it would be practically impossible for the rivals to create a diagnostic program. Therefore, it can be concluded that the software company is the only source of the information that is necessary for maintaining the computer program it creates, and hence it can be said that the first factor is easily fulfilled and weighs in favour of the rivals.

The refusal by the software company can arguably be said to prevent the creation of a new product. This may fall within the second factor. The software company's refusal to license renders the rivals unable to create a diagnostic program and, therefore, unable to enter the software maintenance market for the program written by the software company. However, the ECJ indicated that, to fulfil the second factor, it has to be that a new product is not offered to customers even by the holder of copyright (in this analogous analysis, by the software company). If the software company does not offer the maintenance service itself, it is highly likely that the rivals can argue successfully that the refusal meets the second factor's requirements. But, if the software company does offer the maintenance service or a diagnostic program, it is controversial that the second factor will be fulfilled. It can be argued that the customer's demand is at least supplied by the software company, which makes the refusal more rational than where the software company does not offer the service at all. In addition, the software company can argue that its revenue is derived mainly from software maintenance service because software maintenance accounts for 50-90 percent of total life-cycle costs. It should, therefore, be justified for the company to refuse to disclose the source code information. The argument of the software company also receives support from the judgment of the Privy Council in *Canon Kabushiki Kaisha v. Green Cartridge Company (Hong Kong) Ltd.*.[99] In this case, their Lordships suggested that 'the question of whether it is contrary to the public interest for a manufacturer to be able to exercise monopoly control over his aftermarket cannot usually be answered without some inquiry into the relevant

---

[99] *Canon Kabushiki Kaisha v. Green Cartridge Company (Hong Kong) Ltd.* [1997] FSR 817.

except where the software company does not provide the maintenance service. In the absence of the service, it is quite clear that the exercise of copyright does prevent the emergence of a new product or service and, therefore, such exercise will constitute an abuse.

From the analysis of the application of the *Magill* case to software development and maintenance described above, it can be said that the rivals seem to have difficulties when trying to put their case in the ambit of competition law. The special circumstance which was established in the *Magill* case does not provide an absolute guarantee for the rivals to rely on as a defence to a copyright infringement claim.

Indeed, the High Court and the Court of First Instance (CFI) have in subsequent cases treated the special circumstance in the *Magill* case with caution. In *Philips v Ingman*,[102] Laddie J. stated that '[if] a party is to rely on *Magill* to resist the enforcement of an intellectual property right, it is incumbent on him to plead explicitly what the exceptional features are which take the case outside *Volvo v Veng*.'[103] Likewise, the preliminary ruling of the CFI in *Oscar Bronner v Mediaprint*[104] confirmed that the principle in the *Magill* case must be strictly applied in that it should be given a narrow interpretation. Accordingly, the CFI held that:

> it would still be necessary, for the *Magill* judgment to be effectively relied upon in order to plead the existence of an abuse within the meaning of Article [82] of the Treaty in a situation such as that which forms the subject-matter of the first question, not only that the refusal of the service comprised in home delivery be likely to eliminate all competition in the daily newspaper market on the part of the person requesting the service and that such refusal be incapable of being objectively justified, but also that the service in itself be indispensable to carrying on that person's business, inasmuch as there is no actual or potential substitute in existence for that home-delivery scheme.[105]

---

[101] Ibid.
[102] *Philips v Ingman*, supra note 7.
[103] Ibid., at para 63.
[104] *Oscar Bronner GmbH & Co. KG v Mediaprint Zeitungs und Zeitschriftenverlag GmbH & Co. KG* [1999] 4 C.M.L.R. 112.
[105] Ibid., at para 41.

Therefore, it can be concluded that only in very exceptional circumstances can the rivals rely on competition law to defend themselves against copyright infringement occurred by the process of reverse engineering.

5. ESSENTIAL FACILITY DOCTRINE

The doctrine of "essential facility"[106] has its origins in the United States.[107] It has only once been discussed in the Commission's decision in *Sea Containers/Stena Sealink*.[108] However, this doctrine has found little favour in the ECJ and the CFI. In neither Court has the doctrine ever been considered as a decisive factor in IP cases for determining whether or not an undertaking has abused its dominant position. Rather, the Courts normally look at the requirements set out in Article 82 and progress along the line of such requirements. This is to answer the questions of whether the undertaking in the dispute acquires a dominant position and whether the undertaking abuses its dominant position. In answering these two questions, the Courts do not 'need to import jargon and doctrines from across the Atlantic to deal with the practices of intellectual property owners in market circumstances which make those practices an abuse of a dominant position within the meaning of Article [82]'.[109] Therefore, the "essential facility" doctrine will not be considered in this chapter.

6. NEXUS ISSUES

It is often that a person who infringes intellectual property rights would defend his or her case under competition law by asserting that the owner of intellectual property rights exercises the rights in a way that is prohibited by competition law. The consequence of such a defence is that, if it is successful, the owner of intellectual property rights may not be able to obtain all the relief to which the owner of intellectual property rights would otherwise be entitled. Based on this possibility, this

---

[106] A facility or infrastructure, without access to which competitors cannot provide services to their customers. This definition was given in *Sea Containers/Stena Sealink*, infra note 108 at para 66.
[107] Greaves, supra note 13 at 383.
[108] *Sea Containers/Stena Sealink* [1994] O.J. L 15/8, [1995] 4 C.M.L.R. 84.
[109] Greaves, supra note 13 at 383.

section analyses whether a reverse engineer will be able to rely on competition law to defend a copyright infringement suit brought against him or her.

The possibility of using competition law to invalidate the exercise of copyright is further limited by a requirement that there must be a **nexus** between the copyright infringement claim and the competition law defence, however obvious a software manufacturer is acting in breach of competition law. One of the leading cases on the competition law defence illustrates this point. In *ICI v. Berk Pharmaceuticals*,[110] Megarry V.C. (Vice-Chancellor) was of the opinion that:

> Article 86 prohibits any abuse which falls within the ambit of the Article. Many other acts by the plaintiffs are also prohibited, whether by statute, common law or equity, or under the Treaty. I do not think that it could be said that a person in breach of some statutory or other prohibition thereupon becomes an outlaw, unable to enforce any of his rights against anyone. If the plaintiffs are imposing unfair selling prices in that they charge too much for their product, I cannot see why this breach of the prohibitions of Article 86 means that the defendants are thereby set free from any liability to the plaintiffs if they, the defendants, commit the tort of passing off (or, indeed, any other tort) against them. Nor can I see that a prohibition against the plaintiffs charging unfair prices also amounts to a prohibition against anything that enables them to get a full return from their unfair prices, as by taking steps to prevent competitors from passing off their goods as being the plaintiffs'. Further, if the defendants are right, the ability of the plaintiffs to sue them for passing off will exist and be destroyed in accordance with the prices which for the time being the plaintiffs are charging, coming and going as the plaintiffs charge a fair price or charge a price that is unfairly high or unfairly low, with all the problems that would arise from prices which are in a twilight zone of merely possible or arguable unfairness.[111]

Therefore, Megarry V.C. assented to the plaintiff's contention, namely, the lack of any nexus between the alleged breach of Article 82 (formerly Article 86) and the passing off which was the subject matter of the action.[112]

---

[110] *Imperial Chemical Industries Ltd. v. Berk Pharmaceuticals Ltd.* [1981] 2 C.M.L.R. 91.
[111] Ibid., at 97.
[112] Ibid.

Although the *ICI v. Berk* case was a passing-off case, the principle established by the Vice-Chancellor was approved by the Court of Appeal in a copyright case, namely *British Leyland v. Armstrong*.[113] In this case, Oliver L.J. rejected an argument that relief in a copyright action should be refused because the plaintiffs had entered into agreements with others which were void under Article 81 (formerly Article 85) of the Treaty of Rome. His Lordship reasoned that the fact that the plaintiffs acted contrary to Article 81 of the Treaty had no bearing on whether or not the defendant was entitled to continue to infringe copyright. This, therefore, applies naturally to software copyright infringement cases. A reverse engineer, it is submitted, may not continue his or her reverse engineering activities by simply asserting that a software manufacturer has acted in breach of competition law; a **nexus** is required. This begs the question: what is the required nexus?

In *Volvo v. Veng*,[114] the European Court of Justice explained what was the required nexus in a copyright infringement case whereby the defendant relied on a competition law defence under Article 82.[115] The European Court stated that:

> It must however be noted that the exercise of an exclusive right by the proprietor of a registered design in respect of car body panels may be prohibited by Article [82] if it involves, on the part of an undertaking holding a dominant position, certain abusive conduct such as the arbitrary refusal to supply spare parts to independent repairers, the fixing of prices for spare parts at an unfair level, or a decision no longer to produce spare parts for a particular model even though many cars of that model are still in circulation, provided that such conduct is liable to affect trade between member-States.[116]

It is submitted that the explanation of the European Court is somewhat inconsistent with the Vice-Chancellor's principle in *ICI v. Berk* and with the opinion of Oliver L.J. in *British Leyland v. Armstrong*. In *ICI v. Berk* and *British Leyland v. Armstrong*, the plaintiffs had already or presumably been in breach of competition law but the defendants were still unable to raise the competition law defence. This was because

---

[113] *British Leyland Motor Corporation Ltd. v. Armstrong Patents Co. Ltd.* [1984] 3 C.M.L.R. 102.
[114] *AB Volvo v. Eric Veng (UK) Ltd.*, supra note 61.
[115] Aldous J. (in *Ransburg-Gema AG v. Electrostatic* [1989] 2 C.M.L.R. 712, 717) expressed that paragraph 9 of the European Court of Justice in the *Volvo v. Veng* case pointed to what was the nexus that was required for Article 82 to provide a defence.
[116] *Volvo v. Veng*, supra note 61, at para. 9.

there was no nexus between the plaintiffs' breach of competition law and the grant of the relief which the plaintiffs claimed. In *Volvo v. Veng*, on the other hand, the defendant was not successful in raising a competition law defence because the plaintiff did not act in breach of competition law. The European Court, however, suggested that a nexus could have been established if there were a breach of competition law by the plaintiff. This is because the European Court stated that the exercise of an exclusive right would be prohibited if there were practices contrary to Article 82, such as those given in the passage cited above. This means that the defendant would be able to continue his or her infringing act if the plaintiff exercises the exclusive rights in the way that is contrary to the provisions of competition law. For this reason there is a nexus for the defendant to raise a competition law defence.

According to judgments discussed above, it can be seen at this stage that there are two different opinions of the well-respected authorities in deciding whether the nexus between the infringement claim and the competition law defence exists. The first opinion comes from the Court of Appeal in *British Leyland v. Armstrong* which confirms the approach of Megarry V.C. that a breach of competition law by the plaintiff does not automatically equip the defendant with a defence against a copyright infringement claim. Therefore, a nexus does not sufficiently exist simply because the plaintiff acts in breach of competition law. In contrast, the second opinion coming from the European Court of Justice in *Volvo v. Veng* makes it clear that the exercise of exclusive rights may be invalid or prohibited if the plaintiff has acted in breach of competition law. As a consequence, a nexus exists when the exercise of exclusive rights involves a breach of competition law. The first opinion, if applied to software copyright infringement cases, will effectively leave a defendant reverse engineer with no defence against a plaintiff software manufacturer. All kind of reverse engineering activities are likely to be prohibited. The only sensible recourse in this circumstance is to implement this thesis' proposed framework to allow reverse engineering activities. The second opinion, on the other hand, will leave reverse engineers with some possibilities for arguing against a copyright infringement caused by the process of reverse engineering. A reverse engineer will be able to prevent the software manufacturer from enforcing the exclusive rights if the software manufacturer's exercise of the exclusive rights falls foul of competition law. Under this circumstance, using competition law is an alternative to the thesis' proposed

framework, provided that the reverse engineer can meet the stringent conditions established in the *Magill* case.[117]

A survey and analysis of case law show that the courts in subsequent cases prefer the second opinion to the first opinion but this does not mean that the first opinion is completely disregarded. Three High Court cases illustrate the position that the second opinion is preferred. Aldous J. in *Quantel v. Electronic Graphics*[118] contemplated that a nexus existed when he stated that 'there [was] no issue to be tried unless the defendants [could] plead and prove breaches of Article [82]'.[119] Similarly, it can be inferred from the suggestion of the High Court in *Yale Security Products v. Newman*[120] that a nexus is considered to exist when the plaintiffs abuse their dominant position. The High Court stated:

> The plaintiff probably did not have a dominant position in the market at the time the case was before the court, but if the action were to succeed and the trade to take notice of it, the plaintiff would presumably have a dominant position, and it was arguable that that was sufficient **to establish abuse of a dominant position as a defence.**[121] **(This author's emphasis)**

Another High Court case, which confirms that a nexus between the alleged infringement of intellectual property rights and the competition law defence can be said to exist where the plaintiffs have been in breach of competition law, is *Pitney Bowes v. Francotyp-Postalia.*[122] Hoffmann J. in this case was facing the same question: whether or not a nexus existed. Although he arrived at the same conclusion as the two previous authorities cited above, he referred instead to a European Court's judgment in *Parke, Davis & Co. v. Probel.*[123] In this case, it was held that Article 82 could affect the rights of the holder of patents if the use of the patent were to degenerate into an abuse of a dominant position. Although he noted that the *Parke, Davis & Co. v. Probel* case gave little guidance, he accepted that this provided for the

---

[117] See discussion in section 4.4 of this chapter.
[118] *Quantel Limited v. Electronic Graphics Limited* [1990] R.P.C. 272.
[119] Ibid., at 276. Although this case is a patent case but the defence based on competition law equally applies to a copyright case because both patent and copyright are a form of industrial and commercial property with in the meaning of the provisions of the Treaty of Rome.
[120] *Yale Security Products Limited. v. Newman* [1990] FSR 320.
[121] Ibid., at 322, also at 323.
[122] *Pitney Bowes Inc. v. Francotyp-Postalia GmbH and Another* [1991] FSR 72.

possibility of a defence under Article 82.[124] Hoffmann J. indeed investigated further into English case law and found that the Court of Appeal in *British Leyland v. T.I. Silencers* did decide that the allegation of an abuse of a dominant position could arguably constitute a defence. This authority persuaded Hoffmann J. to hold that there was a causal connection if there was in actual fact a breach of competition law by the plaintiff.[125]

The consistency of the approach adopted by the High Court in those three cases was subsequently interrupted by a judgment of the Court of Appeal which resurrected the first opinion. In *Chiron v. Murex*,[126] Staughton L.J. stated clearly that:

> [e]ven if it be shown that Chiron (i) have a dominant position in a relevant market, and (ii) are abusing that position, (iii) so as to affect trade between Member States, in my opinion that does not in the present case lead to the conclusion that a national court should refuse to enforce Chiron's patent.[127]

In addition, Balcombe L.J. in the same case expressed his concern along the line with that of Staughton L.J.. Balcombe L.J. was of the opinion as follows:

> To suggest that there is a sufficient nexus between the abuse and the relief sought is to make the plaintiffs outlaws unable to enforce their rights against anyone and to leave Murex free to continue infringement of Chiron's patent. I find this an attractive submission which undoubtedly has some support in the English authorities.[128]

However, Staughton L.J. concluded his judgment by stating that 'there [might] be extraordinary cases, where the holder of a patent should be refused relief against an infringer on the ground that he [was] in breach of Article [82]'.[129] This leaves a room for the courts in subsequent cases to find that a nexus can exist in special circumstances. As a result of the Court of Appeal's judgment in this case, it can be

---

[123] *Parke, Davis & Co. v. Probel* [1968] E.C.R. 55.
[124] Ibid., at 76.
[125] Ibid.
[126] *Chiron Corporation v. Murex Diagnostics Ltd. (No. 2)* [1994] FSR 187.
[127] Ibid., at 199.
[128] Ibid., at 197.
[129] Ibid., at 200.

concluded that not every kind of abuse of a dominant position can be relied on by the rivals as a defence to copyright infringement cases.

7. CONCLUSION

From the discussion above, it is clear that the possible use of competition law as a defence to copyright infringement caused by the process of reverse engineering or as a mechanism to legalise reverse engineering is not widely open for the rivals. In relying on Article 81, the main difficulty is that usually there is no collusion between undertakings which would bring the Article into play. The only collusion which may occur is the potential collusion between the software company and the rivals in the form of software licensing.

However, the illegal licensing may not be used as a defence to copyright infringement because it does not always have a connection to the infringement claimed, i.e. no nexus between the claim of copyright infringement and the competition law defence. Therefore, it can be said that the rivals would not easily rely on Article 81 as a defence. Likewise, the same can be said to happen in their reliance on Article 82.

As it has been established that the mere possession of copyright does not automatically imply a dominant position, the rivals need to prove such a dominant position by referring to the product market. The inconsistency in the Court's approach for determining the product market, however, creates uncertainty which will not benefit the rivals in relying on the competition law defence.

Moreover, the approach which considers the substitutability of the product on both the demand and the supply sides will make the establishment of a dominant position more difficult. This is because it will widen the market in question and the software company will be less likely to be proved dominant in that market.

Even if a dominant position can be found, the abuse of such a position has been confined to some special circumstances only. The special circumstance established in the *Magill* case has been interpreted very narrowly in subsequent cases. This means

that only in an exceptional case will the rivals be able to rely on it. Furthermore, the need of nexus between the competition law defence and the copyright infringement claim will further reduce the possibility of relying on competition law. Therefore, most of the final aims of the process of reverse engineering will not be allowed by exercising competition law as they will not meet the criteria of the special circumstance. Hence, the commercial problems raised in chapter two[130] will not be readily solved by using competition law to legalise reverse engineering. It can be concluded that, for this reason, the proposed framework is needed to solve such commercial problems and to stimulate the advancement of technology.

---

[130] See the discussion in section 5 of Chapter 2. In brief, these commercial problems are inability to develop interoperable products, inability to develop competing products, inability to develop other products, inability to perform perfective maintenance (together with the enhancement of software), inability to perform corrective maintenance, inability to perform adaptive maintenance and inability to perform preventive maintenance.

CHAPTER EIGHT

CONCLUSION

This thesis has suggested that reverse engineering should be legalised and the determination of infringement should be centred on the comparison of the two finished products. It is submitted that this will create a more open creative environment for the exploitation of ideas, and stimulate greater encouragement of competition in the market. The path leading to this thesis' suggestion is summarised as follows. It is also illustrated in the form of a flowchart at the end of this chapter.

Reverse engineering is a dynamic area in the software engineering field. It has received attention from both academia and industries around the world. This can be seen from the fact that Working Conference on Reverse Engineering (WCRE)[1] has been held every year to discuss and explore the advancement of software reverse engineering technology. As shown in Chapter two, the technology of reverse engineering and the practice of it have been applied to both software development and software maintenance, not only by the proprietor of software but also by third parties, because the information retrieved by means of reverse engineering can then be used for many purposes.[2]

However, copyright protection given to computer programs renders the practice of reverse engineering by third parties illegal.[3] Such a practice by third parties will in effect result in an infringement of copyright of the owner of computer programs on two grounds. These are infringement of copyright by copying and making adaptation of the original program because during the process of reverse engineering unauthorised copies of the original program will be created and the *pseudo* version of derived source code will amount to an adaptation of the original program.[4]

---

[1] Sponsored by the IEEE Computer Society: Technical Council on Software Engineering, <http://reengineer.org/wcre/> accessed on 24/02/01.
[2] See the discussion in section 3 of Chapter two.
[3] See the discussion in section 4 of Chapter two.

The prohibition of third parties' practice of reverse engineering will lead to many commercial problems.[5] In software development, these problems are: inability to develop interoperable products, inability to develop competing products and inability to develop other products, which are based on the same ideas but which do not compete or are not interoperable with the original product. In software maintenance, the problems are: inability to modify or enhance the existing products, inability to correct error, inability to adapt it for a change of the supporting platform and inability to take actions necessary to make subsequent maintenance of software in question more efficient and reliable.

Although these problems mentioned above are the third parties' and not the proprietor of software's, the problems will affect the whole of society. This is because, firstly, competition will be reduced as the problems create a barrier to entry into the market, secondly, an incentive to develop further technology will be reduced as there is little need to do so (customers have already been "locked in" and the proprietor will not lose his or her market shares). This is not in the public's interest.

As scrutinised in Chapter three, at the present time, the existing legal solution does not solve the problems raised above. The provisions of the Software Directive and the Copyright Designs and Patents Act 1988 which permit reverse engineering are ambiguous and impractical, and very limited in scope.[6] Moreover, the common law exception of fair dealing seems not to be able to cope with the problems. Neither does the doctrine of fair use, which has been developed to a great extent in the United States.[7]

This is the very reason why a new solution, the proposed framework suggested by this thesis, is needed. The proposed framework suggests, in Chapter four, that all kinds of reverse engineering activities should be permissible. This suggestion is based on the combination of knowledge of software engineering and the fundamental principle of

---

[4] Ibid.
[5] See the discussion in section 5 of Chapter two.
[6] See the discussion in section 2-3 of Chapter three.
[7] See the discussion in section 4 of Chapter three.

copyright law. This is to separate reverse engineering from forward engineering and consider their legal status on their own merit.[8]

As it has revealed in Chapter two that the purpose of reverse engineering is solely to study the computer program in question and does not involve changing or creating a new system based on the reverse engineered program,[9] the purpose of reverse engineering should never infringe copyright. In contrast, changing or creating a new system based on the reverse engineered program falls within the process of forward engineering. Therefore, it is within this process that infringement should be found.

In theory, forward engineering should be an act of infringement if it aims to produce a substantially similar product.[10] However, in practice, it is not appropriate to determine infringement at this stage because the information added into the creation of the new product can be altered at any time. This will result in uncertainty in determining infringement. Therefore, it is suggested that in determining infringement the court should look at the end product and compare it with the original product.[11] This thesis also suggests the method of comparing the two products as follows.[12]

First, establish the subsistence of copyright in the original program.

1. The court has to consider the program in question as a whole unit.
2. On the issue of originality, the test for the subsistence of copyright, the court has to consider the amount of skill, labour and judgement spent in the creation of the program. In the case of Java programs, it is the amount of skill, labour and judgement in selecting and arranging the objects that has to be considered.
3. Then, the court is to apply Nimmer's factors[13] to filter out non-copyrightable materials. For Java programs, only the second, fourth and fifth factors have to be applied.

---

[8] See the discussion in section 3 of Chapter four.
[9] See the discussion in section 2 of Chapter two.
[10] Substantiality is determined by a qualitative and not a quantitative test.
[11] See the discussion in section 4 of Chapter four.
[12] Full explanation of the suggested method of product comparison can be found in section 5 of Chapter four.
[13] Nimmer's factors are as follows. 1) The mechanical specifications of the computer on which a particular program is intended to run; 2) Compatibility requirements of other programs with which a program is designed to operate in conjunction; 3) Computer manufacturers' design standards; 4)

Second, compare the part which has been copied.

1. In such comparison, the qualitative test must be applied.
2. The court must give no weight to the non-copyrightable part of the original program as the original program has already passed the test of subsistence of copyright.
3. The court must be borne in mind all the time that copyright intends to protect only skill and labour of the author, not the idea and function of software.

There are two main reasons to support the fact that this thesis' suggestion to legalise reverse engineering is a well-thought approach. Firstly, there will be no serious impact on the other branches of intellectual property law, which also give legal protection for computer programs to a certain extent, namely the law of confidence and patent law.[14] Secondly, the proposed framework is more efficient than exercising competition law to legalise reverse engineering (to suppress the exercise of copyright).[15]

With regard to the law of confidence discussed in Chapter five, the legalisation of reverse engineering proposed in this thesis will not destroy the fundamental principle of the law of confidence because the policy of the law of confidence itself shows that it is unwilling to prohibit the study of an object purchased in the open market.[16] The proposed framework will not undermine the confidential relationship in the course of employment either. This is because the proposed framework will not enable the employee or the consultant to assert that their use of confidential information will not come within the "springboard" principle.[17]

Furthermore, should the scope of the law of confidence be extended to cover the situation where the information is acquired by an improper means, the proposed framework will support such an extension. This is because, under the proposed

---

Demand of the industry being serviced; 5) Widely accepted programming practices within the computer industry.

[14] See the discussion in Chapter five and six respectively.

[15] See the discussion in Chapter seven.

[16] See the discussion in section 3 of Chapter five.

[17] See the discussion in section 4 of Chapter five.

framework, a reverse engineered program must be legally purchased or licensed before the process of reverse engineering is commenced.[18]

With respect to patent law discussed in Chapter six, the proposed framework will support the move towards the broadening of the scope of UK patent law to that of the EPC in the light of the EPO's recent decision in the *IBM* case.[19] In this chapter, this thesis also makes a further contribution to knowledge by commentary on the issue of whether or not the proposal to extend software patent protection to the broadest extent (by deleting computer programs from the list of non-patentable subject matters) should be adopted.[20]

It is suggested that such a proposal should not be adopted because it will not benefit the society as a whole. The analysis of the extension shows that the economic theories on which patent law is based will not be met if patent law is to be extended to include all kinds of computer programs. The adverse effect on the public's interest will be a result, for a too-strong monopoly will reduce competition, thus wiping out the incentive of the monopolist to innovate further and giving him a chance to increase the price of software exorbitantly.[21]

If the scope of patent law is not to extend beyond that which is drawn by the EPO, the proposed framework will help fulfil any shortfall on the free flow of information. This is because the proposed framework will prevent the proprietor of software from using copyright law, which automatically grants legal protection to software, to create a *quasi* monopoly, which is not supposed to happen under the copyright regime.

In the analysis of the proposal to extend software patentability to the broadest scope possible, it has been revealed that the utility model may be an appropriate form of protection for computer programs in the future, for it takes a middle approach between copyright law on the one hand, and patent law on the other hand.[22] However, the analysis of its appropriateness and impact on other areas of law is beyond the

---

[18] See the discussion in section 5 of Chapter five.
[19] See the discussion in section 4 of Chapter six.
[20] See the discussion in section 8 of Chapter six.
[21] Ibid.
[22] Ibid.

scope of this thesis. Nevertheless, it is an interesting area and the research into this will provide valuable knowledge to the intellectual property and information technology law fields. Therefore, this thesis strongly recommends this area be further explored.

This thesis concludes with a research into competition law to counter the argument that competition law can be exercised to legalise reverse engineering. In Chapter seven, the research shows that the holder of copyright does not necessary hold a dominant position in the market and the normal exercise of copyright thereof does not create an abuse of a dominant position. Therefore, in general, competition law cannot be asserted to suppress the exercise of copyright law which prohibits the act of reverse engineering.[23]

Although competition law may come into play in a special circumstance such as that in the *Magill* case, the analysis in Chapter seven shows that the special circumstance established in the *Magill* case can hardly be applied to the software industry. Moreover, the requirements of the special circumstance have been given a narrow interpretation in subsequent cases.[24] Thus, this shows the intention of the European Court that it will not readily accept any assertion of competition law to suppress the exercise of copyright law.

In addition, the inconsistency of the approach of the European Court in defining the scope of a dominant position and of the abusive conduct creates legal uncertainty, thereby adding more difficulties to the third party who wishes to rely on competition law as a means to legalise reverse engineering. The research also shows that there is inconsistency of the Court's approach on the issue of nexus.[25] This generates further uncertainty in using competition law to legalise reverse engineering.

Therefore, it is reasonable to conclude that, in general, the exercise of competition law cannot help legalise reverse engineering. Only in exceptional circumstances will competition law come to the rescue, e.g. in the corrective maintenance situation where

---

[23] See the discussion in section 3-4 of Chapter seven.
[24] See *Oscar Bronner GmbH & Co. KG v Mediaprint Zeitungs und Zeitschriftenverlag GmbH & Co. KG* [1999] 4 C.M.L.R. 112.

the software company which creates the program does not anymore offer such a service.

To sum up, the proposed framework does create a new environment which, in theory, endorses the legal principle of copyright law and which, in practice, frees up the ideas and opens the door of free competition.

Hence, it can be said that the proposed framework is needed to solve commercial problems raised in chapter two. Once the proposed framework is established, third parties will be free to develop interoperable and competing products, to the extent that the new products do not infringe copyright of the original product, according to the suggested method of product comparison in chapter four. Moreover, all four categories of software maintenance will be open to all to perform, except the perfective maintenance which results in a new product substantially similar to the original product. This thesis' suggestion will stimulate competition and advancement of technology as well as support the move towards the open system which is receiving recognition from around the world.[26] At the same time, it will confer sufficient legal protection on the manufacturer of computer programs.

By way of summary, the development of this thesis' argument is represented in a flowchart form on the next pages in order to provide the best understanding of the thesis for the reader.

The author hopes that the suggestion made to copyright law will yield benefit to the public as a whole and make a good contribution to the stock of knowledge in information technology and intellectual property laws.

---

[25] See the discussion in section 6 of Chapter seven.
[26] Ed Platt, 'Open source software' The business FT weekend magazine 17.02.01 p. 16.

Flowchart showing the development of the argument.

chapter 2

RE is important and used by the software company and third parties.

RE by 3rd parties is illegal because it infringes copyright.

Prohibition leads to commercial problems both in software development and maintenance.

Although it's a 3rd party's problem, it affects the whole society because 1. reduces competition, 2. no need to develop further technology, 3. not in the public's interest.

chapter 3

Current law doesn't solve the problems – 50B, Art. 6 and fair use are ineffective.

Needs a new solution, i.e. the proposed framework suggested by this thesis.

chapter 4

PPFW combines the knowledge of software engineering and legal principle of copyright law.

This is to separate RE from FE and consider the legal status on their own merits.

RE's purpose is to study, thus, not an infringing process.

FE can be an infringement if aim to produce a substantially similar product.

RE = Reverse Engineering
FE = Forward Engineering
PPFW = Proposed Framework

Not appropriate to determine infringement at FE level as it's impractical.

Need to look at the end product, hence, the proposition of a product comparison test.

PPFW = legalise RE+FE and determine infringement by comparing two end products.

Two reasons to support that this is a good approach
1. no serious impact on the other branch of IP law, 2. better than using competition law to legalise RE.

chapter 5

For law of confidence, legalising RE doesn't destroy foundation of LoC as LoC's unwilling to enforce against the study of an object

It doesn't affect confidential relationship in the course of employment either.

chapter 6

Thesis recommends LoC to extend to cover the acquisition of info. by improper means and PPFW will support this.

For patent law, PPFW'll support the move towards a broader scope of UK patent law to that of the EPC under the EPO's decision.

In such rejection, PPFW will help ensure that info. still flows freely in the area covered by copyright.

Thesis also contributes to the knowledge by giving reasons in rejecting the adoption of the int. proposal to extend EPC to the broadest extent.

LoC = Law of Confidence
EPO = The European Patent Office
EPC = The European Patent Convention 1973

As for competition law, cannot be used to legalise RE Normal exercise of copyright doesn't constitute abuse of a dominant position.

Middle approach may also be taken, i.e. the utility model. Further research recommended.

Competition law will come into play only in exceptional circumstances e.g. in *Magill*. *Magill*'s been given narrow interpretation. Thus, difficult to rely on.

Also difficulties in proving a dominant position and abuse of a dominant position owing to inconsistency in the court's approach. Also inconsistency in proving nexus.

Obstacles mean exercise of competition law has limited use. Only in few cases can it come to the rescue, e.g. corrective maintenance where software company doesn't offer the service anymore.

Therefore, PPFW needed to help solve commercial problems and to give benefit to public as well as to give sufficient protection to the manufacturer of software.

SELECTED BIBLIOGRAPHY

TEXTBOOKS

Anderman, Steven D., *EC Competition Law and Intellectual Property Rights: The Regulation of Innovation* (Clarendon Press: Oxford, 1998).

Bainbridge, David, *Intellectual Property 4th ed.* (Financial Times/Pitman Publishing: London, 1999).

Bainbridge, David, *Introduction to Computer Law* (3rd ed.) (Pitman Publishing: London, 1996).

Bainbridge, David, *Software Copyright Law* 3rd ed. (Butterworths: London, 1997).

Bainbridge, David, *Software Licensing* (Central Law Publishing: Birmingham, 1995).

Birrell, A., *The Law and History of Copyright in Books* (Cassell and Co.: London, 1899).

Campbell, Dennis (ed.), *International Information Technology Law* (John Wiley & Sons: Chichester, 1997).

Carr, Henry and Arnold, Richard, *Computer Software: Legal Protection in the United Kingdom 2nd ed.* (Sweet & Maxwell: London, 1992).

CCTA (Central Computer and Telecommunications Agency) by Richard West, *Reverse Engineering – An Overview* (HMSO: London, 1993).

Cini, Michelle and McGowan, Lee, *Competition Policy in the European Union* (Macmillian Press Ltd: Basingstoke, 1998).

Clifford Chance, *The European Software Directive.*

Copinger and Skone James on Copyright 13th ed. (Sweet & Maxwell: London, 1991).

Copinger and Skone James on Copyright 14th ed. (Sweet & Maxwell: London, 1999).

Cornell, Gary and Horstmann, Cay S., *Core Java* (The Sunsoft Press: California, 1996).

Cornish, W. R., *Intellectual Property 3rd ed.* (Sweet & Maxwell: London, 1996).

Cornish, W. R., *Intellectual Property 4th ed.* (Sweet & Maxwell: London, 1999).

Craig, Paul and Burca, Grainne de, *EC Law: Text, cases, & Materials* (Clarendon Press: Oxford, 1995).

Czarnota, B. and Hart, R., *Legal Protection of Computer Programs in Europe: A Guide to the EC Directive* (Butterworths: London, 1991).

Davies, G., *Copyright and the Public Interest* Vol. 14, IIC Studies (Weinheim, VCH VERlag, 1994).

Faull, Jonathan and Nikpay, Ali, *Fuall & Nikpay: The EC Law of Competition* (Oxford University Press: Oxford, 1999).

Fischer, Stanley and Dornbusch, Rudiger, *Economics* (McGraw-Hill Inc.: New York, 1983).

Forester, Tom and Morrison, Perry, *Computer Ethics 2nd ed.* (The MIT Press: Cambridge, 1994).

Frazer, A. (P.A.V. Hall ed.), Reverse Engineering – hype, hope or here? *Software reuse and reverse engineering in practice* (Chapman & Hall: London, 1992).

Garside, Roger and Mariani, John, *Java: First Contact* (Cambridge, Course Technology, 1998).

Govaere, Inge, *The Use and Abuse of Intellectual Property Rights in E.C. Law* (Sweet & Maxwell, 1996).

Goyder, D.G., *EC Competition Law 2nd ed.* (Clarendon Press: Oxford, 1993).

Gurry, Francis, *Breach of Confidence* (1984).

Halberstam, Simon and Turner, Jonathan DC (eds.), *Countdown to 2000: A Guide to the Legal Issues* (Butterworths: London, 1998).

Harris, Lesley Ellen, *Digital Property: Currency of the 21st Century* (McGraw-Hill Ryerson Limited: Toronto, 1998).

Jackson, Margaret, *The Development of Australian Law to Protect Undisclosed Business Information* (PhD thesis, Law School, The University of Melbourne, 1998).

Korah, Valentine, *An Introductory Guide to EC Competition Law and Practice 6th ed.* (Hart Publishing: Oxford, 1997).

Laddie, Prescott and Vitoria, *The Modern Law of Copyright and Designs 2nd ed.*, (Butterworths: London, 1995).

Liebesny, F., *Mainly on Patents* (Butterworths: London, 1972).

Lloyd, Ian J., *Information Technology Law 2nd ed.* (Butterworths: London, 1997).

Lloyd, Ian J., *Information Technology Law 3rd ed.* (Butterworths: London, 2000).

Machlup, Fritz, *The Political Economy of Monopoly: Business, Labor and Government Policies* (The Johns Hopkins Press: Baltimore, 1952).

Meinhardt, Peter, *Invention Patents & Trade Marks* (Gower Press Ltd.: London, 1971).

Nimmer, Laymond, *The Law of Computer Technology* (Warren, Gorham & Lamont: Boston, 1985).

OECD, *Competition and Trade Policies: Their Interaction* (OECD: Paris, 1984).

Reed, Chris, *Computer Law 3rd ed.* (Blackstone Press Limited: London, 1996).

Reid, Brian C., *Intellectual Property Guides: A Practical Guide to Patent Law 3rd ed.* (Sweet & Maxwell: London, 1999).

Ricketson, Sam, *Intellectual Property: Cases, Materials and Commentary* (Butterworths, Sydney, 1994).

Rowland, Diane and Macdonald, Elizabeth, *Information Technology Law* (Cavendish Publishing Limited: London, 1997).

Saxby, Stephen, *Encyclopaedia of Information Technology Law* (Sweet & Maxwell: London, 1990-2000).

Saxby, Stephen, *Regulation of the Market in Digital Information* (Thesis, 1996).

Saxby, Stephen, *The age of information: the past development and future significance of computing and communications* (Macmillan: London, 1990).

Sterling, J.A.L, *World Copyright Law* (Sweet & Maxwell, 1998).

Stewart, S.M., *International Copyright and Neighbouring Rights* (Butterworths: London, 1983).

Sumpter, Paul, *Copyright and Design* (Butterworths: Auckland, 1996).

Tapper, Colin, *Computer Law 4th ed.* (Longman: London, 1989).

Tritton, Guy, *Intellectual Property in Europe* (Sweet & Maxwell: London, 1996).

West, Richard, *Reverse Engineering – An Overview* (HMSO: London, 1993).

White, T. A. Blanco, *Patents for Inventions and the Protection of Industrial Designs 4th ed.* (Stevens & Sons: London, 1974).

ARTICLES FROM JOURNALS AND WEBSITES

Adcock, Ian, 'Inside story: This is the future' (2001) 635 Auto Express 32.

Anawalt, Howard C., 'High tech - cooperation, competition, or both?' (1991) 6 Santa Clara Computer and High-Technology Law Journal 161.

Bailie, Michael, 'Fully loaded' (2000) 70 Top Gear 154.

Baker & McKenzie, 'EC Legislation – Summary' [1996] 12 The Computer Law and Security Report 249.

Bay, Matteo, 'EC Competition Law and software IPRs. (intellectual property rights)' (1993) 9 Computer Law & Practice 176.

Bently, Lionel and Coulthard, Alan, 'From the Commonplace to the Interface: Five Cases on Unregistered Design Right' (1997) 8 European Intellectual Property Review 401.

Bohm, Nicholas and Ryan and Chris, 'Are Computer Program Patentable?' [1991] Computer Law & Practice 213.
Bull, T.M., Younger, E.J., Bennett, K.H. and Luo, Z.: 'BYLANDS – Reverse Engineering Safety-Critical Systems' Proc International Conference on Software Maintenance, Nice, France, 1995 (IEEE).

Cameron, Euan, 'Lotus v Borland: An Interim for the United Kingdom' (1996) 10 International Review of Law, Computers and Technology 169.

Carolina, Robert, 'Legal Aspects of Software Protection Devices' [1995] 11 The Computer Law and Security Report 188.

Chalton, Simon, 'Implementation of the Software Directive in the United Kingdom: The Effects of the Copyright (Computer Programs) Regulations 1992' [1993] 9 The Computer Law and Security Report 115.

Chan, Carl, 'The Patentability of Software Data Structures After *Lowry* and *Warmerdam*' (1998) 32 New England Law Review 899.

Chikofsky, Elliot J. and Cross II, James H., 'Reverse Engineering and Design Recovery: A Taxonomy' [1990] IEEE Software 13.

Cifuentes, Cristina and Fitzgerald, Anne, 'Reverse Engineering of Computer programs: Comments on the Copyright Law Review Committee's Final Report on Computer Software Protection', (1995) 6 Journal of Law and Information Science 241.

Clapes, Anthony L, 'Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs, (1987) 34 UCLA Law Review 1493.

Clapes, Anthony L., 'Decompilation & European Protection of Computer Programs: A Review of Legal Protection of Computer Programs in Europe: A Guide to the EC Directive by Bridget Czarnota & Robert J. Hart' (1994) 20 Rutgers Computer & Technology Law Journal 329.

Cohler, Charles and Pearson, Hilary, 'Software Interfaces, Intellectual Property, and Competition Policy' (1993) 9 The Computer Law and Security Report 159.

Cohler, Charles B. and Pearson, Hilary E., 'Software interfaces, intellectual property and competition policy. (European Community)' (1994) 16 European Intellectual Property Review 434.

Colombe, Michel and Meyer, Caroline, 'Seeking Interoperability: An Industry Response', (1990) 12 European Intellectual Property Review 79.

Cook, William, 'Statement of William J. Cook before the House Judiciary Committee Courts and Intellectual Property Subcommittee Continued Hearing on the NII Copyright Protection Act – [1996] 12 The Computer Law and Security Report February 1996' [1996] 12 The Computer Law and Security Report 150.

Cornish, W.R., 'Professor Dr. F.K. Beier: an appreciation' (1998) 20 European Intellectual Property Review 199(2).

Costa, Christopher R, 'US v Microsoft II, Department of Justice and Microsoft Prevail on Appeal; US District Court Judge Jackson Approves Consent Decree' [1995] 11 The Computer Law and Security Report 336.

Costa, Christopher R., 'US v Microsoft – US District Court Judge Sporkin Denies Motion to Approve Consent Decree; Department of Justice and Microsoft Appeal' [1995] 11 The Computer Law and Security Report 212.

Davidson, Stephen J., 'Exploring tensions between copyright law and competition' (1997) 14 The Computer Lawyer 1.

Doble, R.G.V., 'Patent and Copyright Protection of Board Games: Do Not Pass Go?' (1997) 10 European Intellectual Property Review 587.

Dorny, Brett, 'Claims–Computer Software' (1997) 10 European Intellectual Property Review D-266.

Doyle, Stephen J., 'GATT, TRIPS, and High Tech' [1995] 11 The Computer Law and Security Report 182.

Dumbill, Eric Alexander, 'EC Directive on Computer Software Protection' [1991] Computer Law & Practice 210.

Durdik, Paul, 'Reverse Engineering As a Fair Use Defense to Software Copyright Infringement' (1994) 34 Jurimetrics Journal 451.

Dyer, D., 'Java decompilers compared' <http://www.javaworld.com/javaworld/jw-07-1997-decommpilers.html> accessed on 25/10/98.

ECIS (UK Implementation Working Group), 'Implementation in UK Legislation of EC Software Directive – The ECIS Response' [1992] 8 The Computer Law and Security Report 11.

ECIS, 'The Proposed EC Directive on the Legal Protection for Computer Programs: Position Statement' [1990] Computer Law & Practice 97.

Evans, Michael, <http://www.hobsonaudley.co.uk/cf14/11_ITLaw2.cfm> accessed on 4/11/00.

Farrell, Joseph, 'Standardization and Intellectual Property', (1989) 30 Jurimetrics Journal 35.

Fenwick & West, 'International Legal Protection for Software' [1996] 12 The Computer Law and Security Report 2.

Fitzgerald, Darren, 'Magill Revisited' [1998] European Intellectual Property Review 154.

Flynn, James, 'Intellectual property and anti-trust: EC attitudes' (1992) 14 European Intellectual Property Review 49(6).

Fox, Eleanor M., 'Trade, competition, and intellectual property - TRIPS and its antitrust counterparts. (Agreement on Trade-Related Aspects of Intellectual Property Rights)(American Association of Law Schools' Intellectual Property Section's Symposium on Compliance with the TRIPS Agreement)' (1996) 29 Vanderbilt Journal of Transnational Law 481.

Franzosi, Mario, 'Patentable Inventions: Technical and Social Phases: Industrial Character and Utility' (1997) 5 European Intellectual Property Review 251.

FriedMan, Lawrence M. and Wojcik, Mark E., 'Computer software as articles of commerce in international trade: the surprising study of Singapore's software subsidies' (1991) 4 Software Law Journal 399.

Gerber, David J., 'Global technological integration, intellectual property rights, and competition law: some introductory comments.(Symposium on Global Competition and Public Policy in an Era of Technological Integration)' (1997) 72 Chicago-Kent Law Review 357.

Gielen, Charles, 'WIPO and Unfair Competition' (1997) 2 European Intellectual Property Review 78.

Gilbert-Macmillan, Kathleen, 'Intellectual Property Law for Reverse Engineering Computer Programs in the European Community' (1993) 9 Computer & High Technology Law Journal 247.

Gomez, Frank C., 'Copyright: preemption - misappropriation. (Washington Post v. Total News, Inc.)(Annual Review of Law and Technology)' (1998) 13 Berkeley Technology Law Journal 21.

Gordon, Mark L., 'Copying to compete: the tension between copyright protection and antitrust policy in recent non-literal computer program copyright infringement cases' (1996) 15 The John Marshall Journal of Computer & Information Law 171.

Gordon, Mark, 'Copying to Compete: The Tension Between Copyright Protection and Antitrust Policy in Recent Non-Literal Computer Program Copyright Infringement Cases' (1996) 15 Journal of Computer & Information Law 171.

Gosling, Paul, 'Who Owns Nature?' (2000) Nov/Dec Accounting & Business 14.

Graham, Lawrence D. and Zerbe, Jr., Richard O. 'Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure' (1996) 22 Rutgers Computer & Technology Law Journal 61.

Greaves, Rosa, "The Herchel Smith Lecture 1998: Article 86 of the E.C. Treaty and Intellectual Property Rights" [1998] EIPR 379.

Green, Nicholas, 'Intellectual property and the abuse of a dominant position under European Union law: existence, exercise and the evaporation of rights. (Symposium: Intellectual Property and Competition Law: Changing Views in the European Community and the United States of America)' (1994) 20 Brooklyn Journal of International Law 141.

Grover, Derrick, 'The Case Against Software Patents' [1996] 12 The Computer Law and Security Report 88.

Grunert, Peter, 'Middle class' (2000) 70 Top Gear 122.

Gunther, Andreas and Wuermeling, Ulrich, 'The German Implementation of the EC Directive on Software Protection' [1993] 9 The Computer Law and Security Report 176.

Hanna, Ramsey, 'Misusing antitrust: the search for functional copyright misuse standards' (1994) 46 Stanford Law Review 401.

Harz, Mercer H., 'Dominance and duty in the European Union: a look through Microsoft Windows at the essential facilities doctrine' (1997) 11 Emory International Law Review 189.

Hayes, Daniel, 'London Branch Report: Reverse Engineering – The American Experience' [1994] Computer and Law 13.

Hayes, David L., 'A Comprehensive Current Analysis of Software "Look and Feel" Protection', <http://www.fenwick.com/pub/lookfeel.html> accessed on 22 July 1998).

Hemnes, Thomas M, 'Three Common Fallacies in the User Interface Copyright Debate, 6 Computer Law & Practice 163.

Holt, Mary Kathryn, 'In search of equilibrium: intellectual property, antitrust, and the personal computer industry' (1988) 2 Software Law Journal 577.

Ignatin, Gary R., 'Let the Hackers Hack: Allowing the Reverse Engineering of Copyrighted Computer Programs to Achieve Compatibility' (1992) 140 University of Pennsylvania Law Review 1999.

Inglis, Andrew and Gringras, Clive, 'Conflict of Intellectual Property Laws: A U.K. Perspective on Europe' (1997) 8 European Intellectual Property Review 396.

Irlam, Gordon and Williams, Ross, 'Negative Correlation of Innovation and Software Patents: Revised version of Appendix D of the League for Programming Freedom's submission to the Patent Office, January 25, 1994' <http://www.base.com/software-patents/scoreboard.html> accessed on 6/12/00.

Jerrard, Don, 'Magill' [1995] 11 The Computer Law and Security Report 151.

Johnson-Laird, Andrew, 'Reverse Engineering of Software: Separating Legal Mythology From Actual Technology' [1992] Software Law Journal 331.

Jones II, Bryce J. and Turner, James R., 'Can an operating system vendor have a duty to aid its competitors?' (1997) 37 Jurimetrics Journal of Law, Science and Technology, Summer 355.

Jones, Gareth, 'Restitution of benefits obtained in breach of another's confidence' (1970) 86 L.Q.R. 463.

Kavanagh, 'Accounting for dot.coms' (2000) Nov/Dec Accounting & Business 38.

Kent, Davey, 'Reverse Engineering of Computer Programs' (1993) 4 Australian Intellectual Property Journal 59.

Kleeman, John, 'Note: Patents for Computer Software? No Thank You' [1996] 12 The Computer Law and Security Report 36.

Kobak Jr., James B., 'Running the gauntlet: antitrust and intellectual property pitfalls on the two sides of the Atlantic' (1996) 64 Antitrust Law Journal 341.

Korah, Valentine, 'Commentary. (Symposium: Intellectual Property and Competition Law: Changing Views in the European Community and the United States of America)' (1994) 20 Brooklyn Journal of International Law 161.

Lai, Stanley, 'Recent Developments in Copyright Protection and Software Reverse Engineering in Singapore: A Triumph for the Ultra-protectionists?' [1997] 9 European Intellectual Property Review 525.

Lande, Robert H. and Sobin, Sturgis M., 'Reverse engineering of computer software and U.S. antitrust law.(Symposium: High Technology, Antitrust & the Regulation of Competition)' (1996) 9 Harvard Journal of Law & Technology 237.

Laurie, Ronald S., McCutchen, Doyle, 'Comparison of Patent and Copyright Protection for Computer Software under US Law' [1996] 12 The Computer Law and Security Report 80.

Legal Advisory Board, 'The EC Legal Advisory Board's Reply to the Green Paper on Copyright and Related Rights in the Information Society' [1996] 12 The Computer Law and Security Report 143.

Lehmann, M. and Dreier, T., 'The Legal Protection of Computer Programs: Certain Aspects of the Proposal for an (EC) Council Directive' [1990] Computer Law & Practice 92.

Lewis, Andrew, 'Windows 95 Block Internet Rivals' [1996] 12 The Computer Law and Security Report 129.

Maebius, Stephen B., 'The New Use of Fair Use: Accessing Copyrighted Programs Through Reverse Engineering' (1993) 75 Journal of the Patent and Trademark Office Society 431.

Mahmoud, Qusay H., 'Java Tip 22: Protect Your Bytecodes from Reverse Engineering/Decompilation' <http://www.javaworld.com/javaworld/javatips/jw~javatip22.html> accessed on 25/10/98.

Marchal, Benoît and Meurrens, Marc, 'Java Decompilation and Reverse Engineering: Part I' <http://www.javacats.com/US/articles/decompiler1.html> accessed on 3/11/98).

McGowan, David, 'Regulating competition in the information age: computer software as an essential facility under the Sherman Act.(Eighth Annual Computer Law Symposium: Recent Developments in Computer Law)' (1996) 18 Hastings Communications and Entertainment Law Journal 771.

Meaden, Richard, 'Porsche 911 Carrera 4' (1998) 2 EVO 68.

Miller, Clifford G., 'Magill: 'essential facilities' doctrine arrives in the EC' (1995) 11 Computer Law & Practice 65.

Mills, James, 'Jag's new R-types' (2000) 573 Auto Express 26.

Mills, John G., 'Possible defenses to complaints for copyright infringement and reverse engineering of computer software: implications for antitrust and I.P. law' (1998) 80 Journal of the Patent and Trademark Office Society 101.

Morgan, Nathan, 'Road test new BMW 5-Series'/ Mercedes E-Class (2001) 634 Auto Express 34.

Morrison, Linda G., 'The EC Directive on the Legal Protection of Computer Programs: Does It Leave Room for Reverse Engineering Beyond the Need for Interoperability?', 25 Vanderbilt Journal of Transnational Law 293.

Müller, H., 'Understanding Software Systems Using Reverse Engineering Technologies Research and Practice' <http://www.rigi.csc.uvic.ca/UVicRevTut/Abstract.html> accessed on 29/10/98.

Newman, Jonathan, 'The Patentability of Computer-related Inventions in Europe' (1997) 10 European Intellectual Property Review 701.

News from CNET News.com on 2 December 1999 <http://news.cnet.com/news/0-1007-200-1476392.html?pt.salon> accessed on 6/12/00.

Owens, Jonathan, 'Software Reverse Engineering and Clean-Rooming, When Is It Infringement?' (1993) 9 Computer & High Technology Law Journal 527.

Platt, Ed, 'Open source software' The business FT weekend magazine 17.02.01 p. 16.

Powell, Mark, 'Drafting Software Licences in Light of the EEC Competition Rules' (1993) 9 The Computer Law and Security Report 254.

Reed, Chris, 'Reverse Engineering Computer Programs without Infringing Copyright', (1991) 2 European Intellectual Property Review 47.

Reichman, J.H., 'Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research' (1989) 42 Vand. L. Rev. 639.

Rhodes, Michael, 'Prohibiting the Decompilation of Software' (1993) 9 Computer Law & Practice 1.

Ruping, Karl, 'Copyright and an integrated European market: conflicts with free movement of goods, competition law, and national discrimination' (1997) 11 Temple International and Comparative Law Journal 1.

Sanz, Monica Esteve, 'The Magill judgment - its consequences for the software industry. (European Community)' (1995) 11 Computer Law & Practice 67.

Saxby, Stephen, 'Fujitsu Patent Judgement Issued' [1996] 12 The Computer Law and Security Report 321.

Saxby, Stephen, 'Microsoft Consent Decree Rejected' [1995] 11 The Computer Law and Security Report 166.

Saxby, Stephen, 'Netscape and Microsoft Dispute Pricing' [1996] 12 The Computer Law and Security Report 404.

Saxby, Stephen, 'Patent Office Must Respond to Competitive Challenge, Says Ian Taylor' [1996] 12 The Computer Law and Security Report 254.

Saxby, Stephen, 'Patent Office Seeks Views on Programme Formats' [1996] 12 The Computer Law and Security Report 251.

Saxby, Stephen, 'Settlement Proposed in Pentium Chip Dispute' [1995] 11 The Computer Law and Security Report 226.

Saxby, Stephen, 'Telecom 95 Breaks New Ground' [1996] 12 The Computer Law and Security Report 63.

Saxby, Stephen, 'US Court of Appeals Approves Microsoft Consent Decree' [1995] 11 The Computer Law and Security Report 283.

Saxby, Stephen, 'Microsoft Antitrust Settlement to be Challenged' [1995] 11 The Computer Law and Security Report 105.

Shulman, Seth, 'Software Patents Tangle the Web' (2000) March/April Technology Review <http://www.techreview.com/articles/ma00/shulman.htm> accessed on 6/12/00.

Soma, John T., Winfield, Gus and Friesen, Letty, 'Software Interoperability and reverse engineering' [1994] 20 Rutgers Computer & Technology Law Journal 189.

Sterling, J.A.L., 'Testing for Subsistence and Infringement of Copyright in Computer Programs: Some US and UK Cases' [1995] 11 The Computer Law and Security Report 119.

Stewart, Tom, 'In bed with Modena' (1999) 69 Top Gear 96.

Teter, Timothy S., 'Merger and the Machines: An Analysis of the Pro-Compatibility Trend in Computer Software Copyright Cases' (1993) 45 Stanford Law Review 1061.

The Bureau of National Affairs Inc., Washington D.C., 'Bill Would Tighten PTO Procedure For Issuing Business Method Patents' (2000) 1 Computer Technology Law Report 198 (Number 11, 6 October 2000).

The OTA Report – Background Paper – Summary, Overview and Issues' [1990-91] 4 The Computer Law and Security Report 11.

The University of Queensland, 'The General Approach to Decompilation' <http://www.csee.uq.edu.au/csm/decompilation/general.html> accessed on 26/10/98.

Tocups, Nora M. and . O'Connell, Robert J, 'Patent Protection for Computer Software' (1997) 14 The Computer Lawyer 18.

Vermut, Richard S., 'A Synthesis of the Intellectual Property and Antitrust Laws: A Look at Refusals to License Computer Software' (1997) 22 Columbia-VLA Journal of Law & the Arts 27.

Wallberg, John, 'Scenario from Ethics and Reverse Engineering' (last updated: 5 September 1998) <http://ethics.cwru.edu/projects/rev_scen.html> accessed on 29/10/98.

Wiebe, Andreas, 'European Copyright Protection of Software from a German Perspective' (1993) 9 Computer Law & Practice 79

William, John A., 'Can Reverse Engineering of Software Ever Be Fair Use? Application of Cambell's "Transformative Use" Concept' (1996) 71 Washington Law Review 255.

OFFICIAL DOCUMENTS

Act Revision of the Convention on the Grant of European Patents (Munich, 29 November 2000), MR/3/00 Rev. 1 e, <http://www.european-patent-office.org/epo/dipl_conf/pdf/em00003a.pdf> accessed on 10/12/00.

Amended proposal for a European Parliament and Council Directive approximating the legal arrangements for the protection of inventions by utility model, (COM(1999) 309 final/2) 12.07.1999, (97/0356 (COD)).

Basic proposal for the revision of the European Patent Convention, done at Munich 13/10/2000, MR/2/00.

CLRC Report on Computer Software Protection, Chapter 10. <http://www.agps.gov.au/customer/agd/clrc/clrc/word%20comp%20software%20for%20ws/indexCS.html> accessed on 14/6/99.

Commission Proposal for a Council Directive on the Legal Protection of Computer Programs, COM (88) 816 final, 1989 O.J. (C91) 4.

http://www.delphion.com/cgi-bin/viewpat.cmd/US05870717__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05963924__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05987140__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06016484__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06026379__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06026430__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06061726__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06101485__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06119105__

http://www.delphion.com/cgi-bin/viewpat.cmd/US06138119__

Promoting innovation through patents – The follow-up to the Green Paper on the Community Patent and the Patent System in Europe, (COM(1999)42), para. 1.3, <http://www.europa.eu.int/comm/internal_market/en/intprop/indprop/8682en.pdf> accessed on 11/12/99.

Proposal for a Council Regulation on the Community patent, COM(2000) 412 final, 1.8.2000.

Proposal for a European Parliament and Council Directive approximating the legal arrangements for the protection of inventions by utility model, (COM(97) 691) - part 1

Report by the Committee on legal Affairs and Citizens' Rights A4-0096/99, 25 February 1999.

Report from the Commission to the Council, the European Parliament and the Economic and Social Committee on the implementation and effects of Directive 91/250/EEC on the legal protection of computer programs, COM(2000) 199 final.

Report on the proposal for a European Parliament and Council Directive Approximating the legal arrangements for the protection of inventions by utility model (COM(97)0691-C4-0676/97-97/0356(COD)), Committee on Legal Affairs and Citizens' Rights, Rapporteur: Julio Anoveros Trias de Bes.

Study Contract ETD/99/B5-3000/E/106: The Economic Impact of Patentability of Computer Programs (Report to the European Commission by Robert Hart, Peter Holmes and John Reid, on behalf of Intellectual Property Institute), <http://europa.eu.int/comm/internal_market/en/intprop/indprop/study.pdf> accessed on 21/10/00.

Communication from the Commission to the Council, the European Parliament and the Economic and Social Committee: The follow-up to the Green Paper on the Community Patent and the Patent System in Europe, <http://www.europa.eu.int/comm/internal_market/en/intprop/indprop/8682en.pdf> accessed on 5/10/00.

Conference Resolution, MR/22/00 e, <http://www.european-patent-office.org/epo/dipl_conf/pdf/em00022.pdf> accessed on 10/12/00.

CONTU Report.

Cross, J; Quilici, A; Wills, L; Newcomb, P; and Chikofsky, E, 'Second Working Conference on Reverse Engineering Summary Report' <http://www.cc.gatech.edu/conferences/WCRE95/summary-rpt.html> accessed on 13/11/98.

Decision of the Administrative Council of 24 February 2000 convening a conference of the contracting states to revise the European Patent Convention, Article 1(1). <http://www.european-patent-office.org/epo/ca/e/24_02_00_e.htm> accessed on 18/10/00.

European Commission Green Paper on Copyright [(1) COM (88) 172], [1988] 3 The Computer Law and Security Report 8.

Parliamentary Debates: First Standing Committee on Statutory Instruments, &c., 'Draft Copyright (Computer Programs) regulations 1992' Thursday 10 December 1992 (HMSO, 1992) p. 8-9.

Patents (United States Patent and Trademark Office) Search at <http://www.uspto.gov/patft/index.html> and <http://128.109.179.23/access/search-bool.html> by using keywords "Microsoft" and "program", accessed on 6/12/00. This database is to be discontinued on 31 December 2000.

http://www.delphion.com/cgi-bin/viewpat.cmd/US05710887__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05715314__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05729594__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05754772__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05768385__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05793966__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05797127__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05809144__

http://www.delphion.com/cgi-bin/viewpat.cmd/US05870562__

The amended proposal of 25 July 1999 COM(99) 309.

The CLRC Executive Summary, <http://www.sisa.org.au/SiSASubmission1.html>.

The CLRC Recommendations on Reverse Engineering and Decompilation: Giving Local Developers an Equal Right to Compete (Executive Summary) <http://www.sisa.org.au/SISASubmission1.html> accessed on 25/5/98.

The Competition Act 1998: The Major Provisions OFT 400 March 1999 (Office of Fair Trading).

The Council Directive of 14 May 1991 on the Legal Protection of Computer Programs (91/250/EEC) OJ No L 122/42, 17.5.91.

The Green paper of 1995 COM(95) 370.

The Green Paper on the Community patent and the patent system in Europe COM(97) 314 final, 24.6.97.

The Law Commission (Law Com. No. 110), *Breach of Confidence: Report on a Reference under Section 3(1)(e) of the Law Commissions Act 1965*, presented to Parliament by the Lord High Chancellor, by Command of Her Majesty October 1981 (HMSO: London, 1981).

The Office of Technology Assessment (OTA), *Intellectual Property Rights in an Age of Electronics and Information* (1986).

The Patent Office – Claims to Programs for Computers, <http://www.patent.gov.uk/snews/notices/practice/programs.html> (last updated 10 January 2000) accessed on 13/10/00.

The US Committee's report accompanying the Patent Code, No. 82-1979 at 5 (1952); H.R. Rep. No. 82-1923 at 6 (1952).

WIPO Copyright Treaty, Geneva, December 20, 1996 <http://www.wipo.org/eng/diplconf/distrib/94dc.htm> accessed on 18/10/99.

WIPO, Patent Law Treaty, PT/DC/47, <http://www.wipo.int/eng/iplex/wo_plt.htm> accessed on 06/10/00.

WIPO, Press Release PR/2000/222: Patent Law Treaty is finalised <http://www.wipo.int/eng/pressrel/2000/p222.htm> accessed on 06/10/00.

WTO: Overview: the TRIPs Agreement <http://www.wto.org/english/tratop_e/trips_e/intel2_e.htm#patents> accessed on 18/10/00.

CASES

Alcatel USA, Inc. v DGI Technologies, Inc. Case no. 97-11339 (5th Cir., 1999).

Alfa Laval v Wincanton Engineering [1990] F.S.R. 583

Amazon.com Inc. v. Barnesandnoble.com Inc., 73 F. Supp. 1228, 53 USPQ2d 1115 (W.D. Wash. 1999).

Ansell Rubber Co. Pty. v Allied Rubber Industries Pty. Ltd. [1972] R.P.C. 811.

Apple Computer, Inc. v Franklin Computer Corp. 714 F.2d 1240 (3rd Cir. 1983).

Atari Games Corp. v Nintendo of America, Inc., 18 U.S.P.Q.2d 1935 (N.D.Cal. 1991).

Atari v Nintendo 975 F. 2d 832 (Fed. Cir. 1992).

Baker v Selden 101 U.S. 99 (1879).

Bateman v Mnemonics Inc 79 F.3d 1532 (11th Cir. 1996).

BP Kemi (79/934/EEC) [1979] 3 C.M.L.R. 684.

Bradbury v Hotten (1872) L.R. 8 Ex. 1.

Bradbury v Hotten (1872) LR 8 Exch 1.

British Leyland Motor Corp Ltd v Armstrong Patents Co Ltd [1986] 2 WLR 400.

British Leyland Motor Corp Ltd. v Armstrong Patents Co Ltd. [1986] 2 WLR 400.

British Leyland Motor Corporation Ltd. v. Armstrong Patents Co. Ltd. [1984] 3 C.M.L.R. 102.

British Leyland v Armstrong [1986] RPC 279.

British Leyland v. T.I. Silencers [1981] CMLR 75 (CA).

Canon Kabushiki Kaisha v Green Cartridge Company (Hong Kong) Limited [1997] FSR 817.

Chiron Corporation v. Murex Diagnostics Ltd. (No. 2) [1994] FSR 187.

Commercial Plastics Ltd. v Vincent [1965] 1 Q.B. 623.

Commercial Solvents v. Commission (Case 6&7/73) [1974].

Compco Co. v Daybright Lighting Co. 376 U.S. 234 (1964).

Computer Associates International Inc. v Altai Inc 982 F.2d 693 (2nd Cir. 1992).

Consten and Grundig v. Commission (56&58/64) [1966] E.C.R. 299.

Designers Guild Ltd. v Russell Williams (Textiles) Ltd., Case No: CHANF 1998/0274/3, <http://wood.ccta.gov.uk/courtser/judgements/nsf/> accessed on 04/7/00.

Deutsche Grammophon Gesellschaft mbH v. Metro - SB Grossmarkte GmbH &Co. KG (Case 78/80) [1971] E.C.R. 487.

Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980).

Diamond v. Diehr, 450 U.S. 175, 182 (1981).

Donoghue v Allied Newspapers Ltd. [1938] 1 Ch. 106.

E. Gomme Ltd. v Relaxateze Upholstery Ltd. [1976] R.P.C. 377.

E.I. duPont deNemours & Co. v Christopher 431 F.2d 1012 (5th Cir, 1970).

Electro Cad Australia Pty Ltd. v Mejati RCS SDN BHD [1999] F.S.R. 291 (the High Court of Malaysia – Commercial Division, before Dato' R. Kamalanathan J.).

Europemballage Corporation and Continental Can Company Inc. v. Commission (Case 6/72) [1972] C.M.L.R. D 11, [1973] E.C.R. 215.

Feist Publications Inc v Rural Telephone Service Co. Inc. (1991) S Ct 1282.

Feist Publications, Inc. v Rural Telephone Service Co., 499 U.S. 340 (1991).

Fraser v. Evans [1969] Q.B. 349.

FSS Travel and Leisure Systems Ltd. v Johnson [1999] F.S.R. 505.

Fujitsu Limited's Application [1997] R.P.C 608.

Gale's Application [1991] R.P.C. 305.

Genentech Inc.'s Patent [1987] R.P.C. 553.

Genentech Inc.'s Patent [1989] R.P.C. 147 (C.A.).

Harbor Software, Inc. v Applied Systems, Inc. 925 F. Supp. 1042 (S.D.N.Y., 1996);

Harper & Row Publishers, Inc v Nation Enterprises 471 U.S. 539 (1985).

Hoffman La Roche v Commission (Case 102/77) [1979] 3 C.M.L.R. 211; <http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexplus!prod!CELEXnumdoc&lg=en&numdoc=61976J0085> accessed on 01/08/00.

Hollinrake v Truswell [1894] 3 Ch. 420 (C.A. Chancery Division).

Hollinrake v Truswell [1894] 3 Ch. 420, 427;

Hotel Security Checking Co. v. Lorraine Co., 160 F.467 (2d Cir. 1908).

Hugin [1979] 3 C.M.L.R. 345.

Hustler Magazine, Inc. v Moral Majority, Inc., 769 F.2d 1148, 1152 (9th Cir. 1986).

Ibcos Computer Ltd. v Barclays Mercantile Highland Finance Ltd. [1994] F.S.R 275;

Imperial Chemical Industries Ltd. v Berk Pharmaceuticals Ltd. [1981] 2 C.M.L.R. 91.

Independent Television Publications Ltd v Time Out Ltd. [1984] FSR 64.

J.C. Gleeson v H.R. Denne Ltd. [1975] R.P.C. 471.

Jarrold v Houlston (1857) 3 K & J 708.

Jefferys v Boosey [1854] 4 H.L.C 815 (H.L.).

Joanna Christina Gleeson and Gleeson Shirt Company Ltd. v H.R. Denne Ltd. [1975] RPC 471.

John Richardson Computers Ltd. v Flanders [1993] FSR 497.

Kerrick and Company v Lawrence and Company (1890) 25 QBD 99.

Ladbroke (Football) Ltd. v William Hill (Football) Ltd. [1964] 1 WLR 273.

Lansing Linde Ltd. v Kerr [1991] I.R.L.R. 80.

LB Plastics Ltd v Swish Products Ltd [1979] RPC 551.

Littlewoods Organisation Ltd. v Harris [1977] 1 W.L.R. 1472.

Lotus v. Paperback 740 F Supp 37 (1990).

Mai Systems Corp. v Peak Computer Inc., 991 F. 2d 511, 26 (9th Cir. 1993).

Mars UK Ltd v Teknowledge Ltd.
<http://wood.ccta.gov.uk/courtser/judgments.nsf/93f7e89393ff0e638025655a005738ac/3777d7666c1bcec980256790002b5040?OpenDocument>.

Mazer v Stein 347 U.S. 201 (1954).

Merrill Lynch's Application [1989] R.P.C 561 (C.A.).

MTV Europe v BMG Record (UK) Ltd. [1997] EuLR 100.

Nichols v Universal Pictures Co. 45 F2d. 119 (2d Cir. 1930).

NV Nederlandsche Banden Industrie Michelin v Commission (Case 322/81) [1983] E.C.R. 3461.

Office Angels Limited v Rainer-Thomas [1991] I.R.L.R. 214.

Oscar Bronner GmbH & Co. KG v Mediaprint Zeitungs und Zeitschriftenverlag GmbH & Co. KG [1999] 4 C.M.L.R. 112.

Page v Wisden (1869) 20 LT 435;

Parke, Davis & Co. v. Probel, Centrafarm and others (Case 24/67) [1968] E.C.R. 55.

Philips Electronics N.V. v. Ingman Ltd., CH 1997 P. No. 4100-1 <http://www.courtservice.gov.uk/judgments/judg-frame.htm>.

Pitney Bowes Inc. v Francotyp-Postalia GmbH and Another [1991] FSR 72.

Quantel Limited v Electronic Graphics Limited [1990] R.P.C. 272.

Radio Telefis Eireann (RTE) and Independent Television Publications (ITP) v Commission (Case C-241-42/91 P.) [1995] E.C.R. I-743, [1995] 4 C.M.L.R. 718.

Ransburg-Gema AG v Electrostatic [1989] 2 C.M.L.R. 712.

Re Alappat, 33 F.3d 1526 (Fed. Cir. 1994).

Re Howard, 394 F.2d 869 (Fed. Cir. 1968).

Re IBM's Application, decision of February 1999, Case number: T 0935/97 - 3.5.1; <http://www.european-patent-office.org/dg3/biblio/t970935eu1.htm> accessed on 1/12/99 and the document can be downloaded from <http://www.european-patent-office.org/dg3/pdf/t970935eu1.pdf>.

Re Maucorps, 609 F.2d 481 (Fed. Cir. 1979).

Re Merrill Lynch, Pierce Fenner & Smith Inc's Application [1988] RPC 1.

Re Meyer, 688 F.2d 789 (Fed. Cir. 1982).

Re Schrader, 22 F.3d 290 (Fed. Cir. 1994).

Reuter/BASF (76/743/EEC) [1976] 2 C.M.L.R. D44, para. 35.

Roger Bullivant Ltd. v Ellis [1987] F.S.R. 172.

Rosemont Enterprises, Inc. v Random House, Inc. 366 F.2d 303, 307 (2nd Cir. 1966).

Rot Greeting Cards v United Card Co., 429 F.2d 1106, 1110 (9th Cir. 1970).

Saphena Computing v Allied Collection Agencies (unreported) 3 May 1989.

Schering Chemicals Ltd. v Falkman Ltd. [1982] Q.B. 1 (C.A.).

Scott v Stanford (1867) L.R. 3 Eq. 718.

Sea Containers/Stena Sealink [1994] O.J. L 15/8, [1995] 4 C.M.L.R. 84.

Sears Roebuck & Co. v Stifel Co. 376 U.S. 225 (1964).

Sega v Accolade 977 F. 2d 1510 (9th Cir. 1992).

Smith v Chato (1874) 31 L.T. 77.

Société Technique Minière v Maschinenbau Ulm GmbH [1966] E.C.R. 235 Case 56/65.

Sony Corp. v Universal City Studios, Inc., 464 U.S. 417 (1984).

Sony v Connectix 53 USPQ.2d 1702, 1709 (9th Cir. 2000).

State Street Bank & Trust Co. v Signature Financial Group Inc., 149 F.3d 1368 (Fed. Cir. 1998).

Stenhouse Ltd. v Phillips [1974] A.C. 391.

Stevenson, Jordan & Harrison Ltd. v Macdonald & Evans [1952] 1 T.L.R. 101.

Terrapin Ltd. v Builders' Supply Co. (Hayes) Ltd. [1967] R.R.C. 375.

Tetra Pak International SA v Commission (Case C-333/94 P.) [1996] E.C.R. I-5951.

Total Information Processing Systems Ltd v Daman Ltd [1992] F.S.R 171.

Triplex Safety Glas Co. Ltd. v Scorah (1938) 55 R.P.C. 21.

Twentieth Century Music Corp. v Atiken 422 U.S. 151 (1975).

U.S. v Microsoft (findings of fact) para. 166, at <http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm>.

United Brands Co. (New Jersey, U.S.A.) and United Brands Continental B.V. (Rotterdam) v Commission (Case 27/76) [1978] E.C.R. 207.

United States of America v Microsoft Corporation, Civil Action No. 98-1232 (TPJ), <http://usvms.gpo.gov/conclusions_index.html>.

Vereeniging Van Cemethandelaren v Commission [1973] C.M.L.R. 7. Case 8/72[1] OFT 400.

VICOM/Computer-related invention, Case number T208/84 (OJ 1987, 14), <http://www.law.soton.ac.uk/internal/llb2000-01/year3/infotechlw319/VICOMfulltext.htm>.

Volvo AB v Erik Veng (U.K.) Ltd (Case 238/87) [1989] 4 C.M.L.R. 122.

Wang Laboratories Inc.'s Application [1991] R.P.C. 463.

Wilkins v Aikin (1810) 17 Vesey 422.

Wood Pulp [1993] E.C.R. 1307.

Yale Security Products Ltd. v Newman [1990] FSR 320.

COMPANIES AND ORGANISATIONS (WHICH PROVIDE INFORMATION FOR THIS THESIS)

2600 Magazine <http//www.2600.com> accessed on 26/10/98.

Access Research Corporation (A Division of TYX Corporation), <http://www.accres.com/software.htm> accessed on 26/10/98.

Advanced Software Technologies Inc.<http://www.advancedsw.com/> accessed on 10/11/98.

Amazon (information given by Jeff Bezos, CEO of Amazon) <http://cse.stanford.edu/classes/cs201/projects-99-00/software-patents/amazon.html> accessed on 6/12/00.

Applied Conversion Technologies Inc.<http://www.year2000plus.com/reveng.html> accessed on 10/11/98.

Ascent Logic Corporation <http://www.year2000plus.com/reveng.html> accessed on 10/11/98.

AT&T Labs Research <http://www.research.att.com/sw/tools/reuse/packages/ciao.html> accessed on 16/11/98.

Boeing Company (information given by John Warner, the president of the Boeing Company), <http://www.opengroup.org/comm/case-studies/boeing.htm> accessed on 10/11/98.

BOMARC–Reverse Engineering <http://w3.trib.com/~rollo/bomcat.htm> accessed on 16/11/98.

Brigham Young University <http://www.byu.edu:80/> accessed on 19/11/98.

destopdollars.com, cashsurfers.com, <http://www.get.paid.to.surf-the.web.homepage.com> accessed on 30/01/01.

Fiserv Solutions Inc. <http://www.fisnet.com> accessed on 29/01/01.

Hacker Catalog <http://www.hackerscatalog.com> accessed on 15/12/98.

HSBC Bank <http://www.banking.us.hsbc.com/internetbanking/faqs.asp> accessed on 12/1/01.

IEEE Computer Society: Technical Council on Software Engineering, <http://reengineer.org/wcre/> accessed on 24/02/01.

Imagix Corp. Reverse Engineering Compiler Home Page <http://www.aromatic.com/~cg/rechome.htm> accessed on 17/02/99.

Ipso Facto Consulting, Inc. <http://www.ipsofacto.com/project3.htm> accessed on 10/11/98.

Lloyds TSB
Bank<http://www.infoville.org.uk/banking/online/banking_lloydtsb.htm> accessed on 15/01/01.

Oracle Corporation - Patent Policy
<http://lpf.ai.mit.edu/Patents/testimony/statements/oracle.statement.html> accessed on 3/12/00;

Samba <ftp://samba.anu.edu.au/pub/samba/> accessed on 15/11/98.

Source Retrieval LLC. <http://www.sourceretrieval.com/doccl.html> accessed on 2/11/98.

The Bylands project – Program Transformations,
<http://www.dur.ac.uk/~dcs1ejy/Bylands/transforms.html> accessed on 26/10/98.

The EuroLinux (its petition) <http://petition.eurolinux.org/index_html?LANG=en> accessed on 3/12/00.

The Opengroup Organisation <http://www.opengroup.org/challenge/> accessed on 10/11/98.<http://www.opengroup.org/overview/> accessed on 10/11/98.<http://www.opengroup.org/case-studies/> accessed on 10/11/98.

The Shadow Knights <http://members.tripod.com/~The_Phantom_x/Main.html> accessed on 11/12/98.

The Social Security Administration in Baltimore and Volt Information Sciences in Westbury, Australia, <http://www.accres.com/software.htm> accessed on 26/10/98.

The Underground Railroad <http://www.tomah.com/samarac/> accessed on 30/10/98.

TRUST <http://www.ist-trust.org/trust_frameset.html> accessed on 31/02/01.

University of British Columbia – The Computer Science Department
<http://www.cs.ubc.ca/labs/se/projects/index.html> accessed on 10/11/98.

University of Victoria <http://www.pigi.csc.uvic.ca/UVicRevTut/UVicRevTut.html> accessed on 11/6/99.

Virginia Commonwealth University <http://www.isy.vcu.edu/~paiken/index.htm>,

White Rabbit Technical Services, <http://www.wabasso.com/reverse.htm> accessed on 26/10/98.

WorldNet Bank Inc. <http://www.worldnetbank.com> accessed on 19/12/98.